X/Open Guide

Guide to the Internet Protocol Suite

X/Open Company, Ltd.

X/Open Guide

Guide to the Internet Protocol Suite

ISBN: 1 872630 08 1
X/Open Document Number: XO/GUIDE/91/010

Set in Palatino by X/Open Company Ltd., U.K.
Printed by Maple Press, U.K.
Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.co.uk

Contents

Preface

This Document

The **Guide to the Internet Protocol Suite** is aimed at both implementors and decision makers, and presents a detailed look at the scope of implementations of IPS available on UNIX and derivative systems. It bridges the gap between the official requirements and the reality of current products by presenting current common practice for such implementations. It addresses itself to the following audience:

- network component implementors;
- user IT specialists;
- networked application developers, and
- individuals or organisations responsible for procurement.

Read in conjunction with the host and gateway requirements RFCs (the official statement of requirements for IPS implementations), the guide presents a profile of the protocols and associated functions and options that products currently deliver. The guide reflects the widespread use of public domain versions of many IPS application and service programs (such as Telnet and BIND) by considering the enhanced functionality provided by such software.

In addition, the guide contains information about the probable contents of the networking component of release 4.4 of the Berkeley Standard Distribution (4.4 BSD) currently being produced. This new distribution is likely to be an important influence on future releases of IPS products in the same way as its predecessors. Information about un-released software distribution may not to be reliable; it is included here to assist users in asking informed questions of their IPS suppliers and to help them in assessing answers given.

The structure and scope of the guide follows that of the requirements RFCs:

- RFC-1122, ‘‘Host Requirements - Communications Layers’’;
- RFC-1123, ‘‘Host Requirements - Application and Support’’, and
- RFC-1009, ‘‘Requirements for Internet Gateways’’.

This guide includes these RFCs as an appendix, thus forming a complete reference manual.

Trademarks

The following trademarks are acknowledged:

Ethernet is a registered trademark of Xerox Corporation.

UNIX is a registered trademark of UNIX System Laboratories, Inc. in the U.S.A. and other countries.

X/Open and the ‘‘X’’ device are trademarks of X/Open Company Limited in the U.K. and other countries.

Acknowledgements

The X/Open Company acknowledges the cooperation of the member companies and many other IPS suppliers and organisations in providing information used in the production of this guide. In particular, thanks are due to the Computer Research Group at the University of California for supplying information about their plans for the content and schedule for the 4.4 BSD distribution.

Referenced Documents

The references in this guide are shown as a key in square brackets, e.g. **[HR-COMMS]**. The key may be separated into two parts, separated by a hyphen, in which case the first part specifies a group of related references. Thus in the above example the group ''HR'' refers to the Host Requirements group of references.

Host Requirements RFCs [HR]

[HR-PERS] ''A Perspective on the Host Requirements RFCs'', R. Braden, RFC-1127, October 1989.

[HR-COMMS] ''Requirements for Internet Hosts - Communications Layers'', IETF Host Requirements Working Group, R. Braden, Ed., RFC-1122, October 1989.

[HR-APPS] ''Requirements for Internet Hosts - Application and Support'', IETF Host Requirements Working Group, R. Braden, Ed., RFC-1123, October 1989.

General [GEN]

[GEN-IAB] ''Internet Activities Board'', V. Cerf, RFC-1160, May 1990.

[GEN-PROTO] ''IAB Official Protocol Standards'', J. Postel, RFC-1140, May 1990.

[GEN-PCIS] ''Protocols for X/Open PC Interworking: SMB'', Developers' Specification, X/Open Company Limited, 1991.

[GEN-NB1] ''Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and Methods'', DARPA, IAB, End-to-End Services Task Force, NetBIOS Working Group, RFC-1001, March 1987.

[GEN-NB2] ''Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications'', DARPA, IAB, End-to End Services Task Force, NetBIOS Working Group, RFC-1002, March 1987.

Physical Layer [PH]

[PH-ENET1] ''The Ethernet - A Local Area Network'', Version 1.0, DIX, September 1980.

[PH-ENET2] ''The Ethernet - A Local Area Network'', Version 2.0, DIX.

[PH-802.3] ''IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications'', IEEE, IEEE Std 802.3-1985, 1985.

Link Layer [LK]

[LK-ARP] ''An Ethernet Address Resolution Protocol'', D. Plummer, RFC-826, November 1982.

[LK-TRAIL] ''Trailer Encapsulations'', S. Leffler and N, Karels, RFC-893, April 1984.

[LK-ENET] ''A Standard for the Transmission of IP Datagrams over Ethernet Networks'', C. Hornig, RFC-894, April 1984.

[LK-802] ''A Standard for the Transmission of IP Datagrams over IEEE 802 Networks'', J. Postel and J Reynolds, RFC-1042, February 1988.

[LK-802.2] ''IEEE Standards for Local Area Networks: Logical Link Control'' IEEE, IEEE Std 802.2-1985, 1985.

[LK-SNAP] ''IEEE Standards for Local Area Networks: Overview and Architecture'', IEEE, IEEE Std 802.1a (Draft), 1989.

Internet Layer [IP]

[IL-IP] ''Internet Protocol (IP)'', J. Postel, RFC-791, September 1981.

[IL-ICMP] ''Internet Control Message Protocol (ICMP)'', J. Postel, RFC-792, September 1981.

[IL-SUB] ''Internet Standard Subnetting Procedure'', J. Mogul and J. Postel, RFC-950, August 1985.

Transport Layer [TL]

[TL-UDP] ''User Datagram Protocol'', J. Postel, RFC-768, August 1980.

[TL-TCP] ''Transmission Control Protocol'', J. Postel, RFC-793, September 1981.

[TL-SVID] ''System V Interface Definition Issue 2 - VOLUME 3'', AT&T, 1986.

[TL-XTI] ''Revised XTI (X/Open Transport Interface)'', Developers' Specification, X/Open Company Limited, 1990.

Remote Login [RL]

[RL-TNET] ''Telnet Protocol Specification'', J. Postel and J. Reynolds, RFC-854, May 1983.

File Transfer [FT]

[FT-FTP] ''File Transfer Protocol'', J. Postel and J. Reynolds, RFC-959, October 1985.

[FT-TFTP] ''The TFTP Protocol Revision 2'', K. Sollins, RFC-783, June 1981.

Electronic Mail - [EM]

[EM-SMTP] ''Simple Mail Transfer Protocol'', J. Postel, RFC-821, August 1982.

[EM-TEXT] ''Standard for the Format of ARPA Internet Text Messages'', D. Crocker, RFC-822, August 1982.

Support Services [SS]

[SS-DOMC] ''Domain Names - Concepts and Facilities'', P. Mockapetris, RFC-1034, November 1987.

[SS-DOMI] ''Domain Names - Implementation and Specification'', P. Mockapetris, RFC-1035, November 1987.

Gateway Systems [GS]

[GS-GR] ''Requirements for Internet Gateways'', R. Braden and J. Postel, RFC-1009, June 1987.

Chapter 1

Introduction

The X/Open Guide to the Internet Protocol Suite addresses the functionality of implementations of the Internet Protocol Suite for UNIX systems and systems derived from UNIX. The following introductory sections set out the motivation for the guide and define its scope and audience. The main chapters of the guide follow closely the format of the Host Requirements RFCs which form the official statement of requirements for IPS implementations. These RFCs are included as an appendix and the main chapters of this guide should be read in conjunction with the equivalent RFC sections.

1.1 MOTIVATION

1.1.1 Purpose of the Document

Since some vendors started bundling the Internet Protocol Suite (IPS), often popularly referred to as TCP/IP, with their versions of the UNIX operating system, it has become the *de facto* standard for the networking of open systems. Almost all UNIX and derivative systems support an implementation of IPS, either as an integrated part of the operating system or using networking software from an independent supplier.

Although the future of open systems networking lies with Open Systems Interconnection (OSI), the significant investment in existing IPS networks makes it desirable to offer users of such networks some advice on how to make IPS and OSI coexist and how to migrate from IPS to OSI.

X/Open intends to provide this kind of advice in a future X/Open Guide to IPS-OSI Coexistence and Migration. The X/Open Guide to the Internet Protocol Suite has been compiled as a first step towards that other Guide. Its main purpose is to describe the current practice in IPS implementations, in other words the functionality available to most of today's IPS users, as a basis for coexistence and migration.

In practice, this guide presents a profile of the protocols and associated functions and options that current IPS implementations deliver. We therefore hope that purchasers, application writers and network component implementors will also find this book valuable when striving to achieve interoperability of IPS systems and portability of IPS applications.

1.1.2 Background to the Internet Protocol Suite and Internet Standards

The development of the Internet Protocol Suite is closely linked to that of the *Internet* network. The Defence Advanced Research Projects Agency (DARPA) of the U.S. Government supported the development of the TCP and IP protocols to link its ARPANET network with other research networks. The resulting internetwork became known as the Internet and now comprises local, regional and backbone networks which link the government-sponsored research communities of North America and an increasing number of other continents.

Since then, a wide range of commercial IPS products has become available, partly because of the demand for products to connect to the Internet but perhaps also because a

readily re-usable implementation of the suite was distributed with the UNIX-derived 4.3 BSD operating system. As stated above, it has become the *de facto* standard for the networking of open systems and is particularly predominant for the connection of UNIX and derivative systems on a local-area network.

Standards for the Internet Protocol Suite are defined in documents known as Request For Comment notes (RFCs). These documents are issued by a committee, known as the Internet Activities Board (IAB), tasked with coordinating the design, engineering and management of the Internet network. The role of the IAB and its RFCs is itself described in an RFC; currently RFC-1160, ''The Internet Activities Board'' [GEN-IAB]. This committee is empowered by the U.S. Government and by U.S. and international research organisations which fund and use the network. It operates through two subsidiaries:

- the Internet Engineering Task Force, responsible for ''... making the Internet work and for the resolution of all short- and mid-range protocol and architectural issues ...'', and
- the Internet Research Task Force (IRTF), responsible for promoting ''research in networking and the development of new technology, ...''.

The RFCs cover a wide variety of topics, of which protocol descriptions and experience in implementing them are only two. An RFC that represents an IPS standard is qualified by means of a *state*, which defines it as being Proposed Standard, Draft Standard or Standard. In addition, a Standard RFC is assigned a *status* which defines its applicability as being Experimental, Elective, Recommended or Required. For each standard RFC the state and status are themselves defined in an RFC: currently RFC-1140, ''IAB Official Protocol Standards'' [GEN-PROTO].

The development of an IPS protocol is achieved by a process of experimentation: a proposed protocol is implemented and then adopted, refined or rejected in the light of practical experience in using it. This process of development, over a number of years, of the protocol standards has combined with divergent and sometimes incorrect implementations to produce many interworking difficulties. These may be relatively trivial (such as differing interpretations of an ambiguous requirement) or more fundamental (selection of different protocols to implement a particular function). It should be noted that there are no official conformance test suites for the IPS protocols, much final testing being performed at ''connectathon'' events where a wide variety of implementations are brought together to test interworking.

1.1.3 The Host Requirements RFCs

In an attempt to bring order to this situation, the requirements for an implementation of the IPS have been defined by a set of RFCs known as the ''Host Requirements RFCs''. They are:

- RFC-1122 ''Host Requirements - Communications Layers'' [HR-COMMS];
- RFC-1123 ''Requirements for Internet Hosts - Application and Support'' [HR-APPS];
- RFC-1009 ''Requirements for Internet Gateways'' [GS-GR].

These RFCs present a detailed profile defining which protocols and options must be supported by a correct IPS implementation. They refer to other RFCs for the actual

definition of these protocols and options. The requirements RFCs represent the authoritative definition of the requirements for successful interworking of IPS implementations.

The protocol implementor or integrator must however be aware of the motivation for these RFCs. Most of the requirements formalise what is considered current good practice in IPS implementations, while others highlight implementation bugs that have proved disruptive to the operation of the Internet. However, where there is a choice between a number of equally valid implementation decisions and no consensus exists, the requirements RFCs take a position for future harmony. Other requirements are prescriptive of future directions in IPS protocols and are thus not representative of any significant body of current implementations.

1.1.4 Status of the IPS market

In the current market for IPS products, there are unlikely to be any implementations that come close to meeting the Host Requirements in full. In addition suppliers have varying degrees of commitment to updating their products to do so. The Host Requirements RFCs are thus an unreliable guide to what functionality and protocol option choices a particular IPS implementation might deliver. The next version of the Berkeley software 4.4 BSD is likely to go a long way towards full conformance, certainly at the communications layers where considerable efforts are being made to conform to all but the most contentious requirements. Some commercial suppliers of IPS implementations also have schedules for producing conformant products, and it is likely that 4.4 BSD will form the starting point for much of this work. On the other hand, some suppliers have implementations unchanged from the 1986 4.3 BSD source code.

1.1.5 Contents of This Guide

This guide presents what is ''current common practice'' in IPS implementations for UNIX and derivative systems. **Section 1.4**, **Derivation of Current Common Practice**, presents the methodology employed in determining the guide's content.

The guide describes what features an implementation is likely to deliver, and what may reasonably be expected from a remote implementation when interworking. The guide recognises that IPS technology is constantly evolving, giving rise to implementations with a wide range of capabilities. In addition, at the higher layers, public-domain versions of the utilities are readily available providing more advanced features. The guide attempts to reflect the capabilities of such ''state-of-the-art'' implementations.

Finally, to complete the guide, information has been included concerning current work on IPS; assessing the level of conformance to the Host Requirements RFCs that may be expected of future IPS products. It should be noted that this information is provisional, and there is no guarantee that a feature will appear in 4.4 BSD or that it will be supported in any future IPS product. The motivation for including such information here is to assist users in assessing new IPS implementations and asking informed questions of suppliers. The information was kindly supplied by the Computer Research Group at the University of California. The 4.4 BSD distribution is expected to be generally available in the 2nd quarter of 1991. An update to 4.3 BSD, known as ''RENO'', has been distributed containing many of the networking features likely to be part of the new distribution.

1.1.6 Relationship to the Host Requirements RFCs

It is specifically not an objective of this guide to present an alternative standard for new or updated implementations, neither is it intended to endorse bad practice or imply that requirements may be ignored. The Host Requirements RFCs and other specifications that may be issued by the IAB remain the official source of standards for the Internet Protocol Suite.

1.1.7 Structure of the Guide

The main chapters of the guide are presented in a format similar to that of the requirements RFCs. Each section introduces the protocols generally implemented for a particular layer or application and presents a detailed list of the requirements that current products do not meet.

The requirements lists and summary tables in this guide reflect the requirements RFCs in their use of the terms **MUST**, **SHOULD**, **MAY**, **SHOULD NOT** and **MUST NOT** as prescriptive language defining the status of each requirement. The reader is referred to the RFCs [HR-COMMS] and [HR-APPS] for the definitions of the meanings of these terms. **Readers familiar with the use of similar terms in other X/Open publications should beware of differences in meaning.**

1.2 SCOPE

For the purposes of this guide, IPS implementations for UNIX and derivative systems are viewed as belonging to three classes:

- Berkeley implementations, based upon the Berkeley 4.3 BSD or possibly 4.2 BSD code. Access to transport-layer functions is provided via the Socket Interface.
- STREAMS-based implementations, using the System V STREAMS mechanism to build a TCP/IP protocol stack. Access to transport-layer functions is provided via the Transport Layer Interface (TLI).
- Intelligent controller-based implementations, where some or all of the protocol handling is performed on a front-end-processor (FEP).

The systems with which such an IPS implementation might have to interwork can be classified as follows:

- other UNIX and UNIX-derived implementations;
- terminal servers, i.e. LAN terminal concentrators which allow asynchronous terminals connection into an Ethernet-based TCP/IP network;
- personal computers;
- IPS implementations for unrelated operating systems, and
- gateway systems.

The scope of this guide is essentially that of the requirements RFCs upon which it is based. This base scope has been augmented as noted below in several areas considered important to the purpose of the guide. Protocols which are not widely implemented have been excluded. The intention is to include the internet protocols that any networked UNIX or derivative system can be expected to support. The main part of this

guide is organised into three chapters, one for each of the Host Requirements RFCs and one for the Gateway Requirements RFC. The scope of each of these chapters is described in the following subsections.

1.2.1 Communications Protocols Scope

The content of **Chapter 2**, **Communications Protocols**, is founded upon that of the Communications Layers Requirements RFC [HR-COMMS]. The requirements for the main protocols (IP, ICMP, TCP, and UDP) are covered here. At the link layer the requirements relevant to interfacing to Ethernet and IEEE 802.2 LANs are also covered. A small number of implementations provide 802.2 drivers for LANs other than 802.3, however, the level of support is insufficient to qualify as common practice.

Most IPS implementations support TCP/IP over some form of serial connection using the SLIP protocol. The primary use of such links is either to provide connectivity to personal computers or to provide a low-grade inter-LAN link. There is isolated support for connections to X.25 networks, such links are primarily used for inter-LAN links. In either case the IPS implementation is acting as a gateway and so this guide defers consideration of the requirements for interfacing to these networking technologies to **Chapter 4**, **Gateway Systems**. The guide does not consider direct connection of host systems to a WAN as an end-system. The scope has been extended here in two ways. Firstly a section has been included covering requirements at the physical layer. Secondly, **Section 2.5**, **Transport Layer Interface**, has been added to specify the requirements for an API to the services provided by the transport protocols, specifying the X/Open XTI specification.

The RARP protocol, used by hosts which obtain configuration information across the network at boot time, is excluded from this section because it is neither an official protocol as defined by the requirements RFCs nor is it widely implemented by systems that this guide addresses.

It is possible to use the IPS transport protocols to provide a transport service for PC interworking applications. The X/Open Developers' Specification ''Protocols for X/Open PC Interworking: SMB'' [GEN-PCIS] endorses the RFCs ''Protocol Standard for a NetBIOS service on a TCP/UDP transport: ...'' [GEN-NB1] and [GEN-NB2] for the use of SMB protocol in the IPS environment. However, due to the currently limited number of implementations, this cannot be considered common practice. Consequently it is not considered further by this guide.

1.2.2 Applications and Support Scope

The content of **Chapter 3**, **Applications and Support**, is based upon that of the Applications and Support Requirements RFC [HR-APPS]. The requirements for the basic applications protocols (Telnet, FTP, TFTP and MAIL) and the name translation service (DNS) are covered here.

An important class of IPS applications excluded are those known as the Berkeley R-Protocols, distributed with the Berkeley 4 BSD distributions and widely supported by IPS implementations in general. These protocols are excluded from the scope of this guide because they are not covered by the Host Requirements documents and because no formal definition of the protocols' operation exists. However, it is recognised that no implementation of IPS for a UNIX or derivative system would be complete without them.

The requirements RFC [HR-APPS] restricts its scope to the basic application-level protocols provided by an IPS implementation. Thus it excludes some important classes of higher-level applications, this guide reflects the scope of [HR-APPS]. In particular applications providing networked file access are excluded.

[HR-APPS] covers two subjects (Remote Booting and Remote Management) which are excluded from this guide. In the case of remote booting, the BOOTP protocol is used in a number of network applications such as X-terminals and PC networking for obtaining configuration details prior to down-loading programs. However the protocol is not widely implemented by host systems and is consequently excluded. The Remote Management section of the requirement RFC acknowledges that two protocols currently exist (SMTP and CMOT) and that few implementations are at production status. It includes no formal requirements other than that future products should provide an implementation of one or other of the protocols, the subject is thus outside the scope of this guide.

1.2.3 Gateway Systems Scope

The content of **Chapter 4**, **Gateway Systems**, is founded upon the Gateway Requirements RFC [GS-GR]. It covers requirements for IP and ICMP which are specific to gateway subsystems, routing protocols and requirements for interfacing to the wider range of network technologies which gateway systems support.

The scope of **Chapter 4** is determined by the way in which UNIX and UNIX-derived systems participate in wide-area internet networks. Generally, the system is connected to a local-area network (LAN), usually an Ethernet, as part of an autonomous system which is connected to a wide-area network (WAN) such as the DARPA Internet. Consequently, the system participates in the network primarily as a host. The system may perform the role of a gateway between two or more LANs and occasionally provides the gateway from a group of interconnecting LANs into the WAN. As a result of this perception of a system's roles, this chapter excludes consideration of the requirements for connecting a host directly to a WAN and for operation as a gateway within the WAN.

1.3 AUDIENCE

A number of groups of readers were considered in the production of this guide, as described below.

1.3.1 IPS Users

For user IT specialists and those responsible for procurement the guide represents a profile of IPS implementations for UNIX and derivative systems (the term *profile* is used in the sense defined by the ISO protocols, see **Appendix A**, **Glossary**, for details). It can be used as a benchmark for assessing and comparing products. For these readers all chapters of the guide contain useful information; however, **Chapter 4**, **Gateway Systems**, may be skipped where the system being considered is purely a host system.

1.3.2 Networked Application Developers

The networked application developer may use this guide to help in the complex task of selecting the sub-set of IPS features which can safely be used in an application which is to be portable. For these readers, **Chapter 3**, **Applications and Support**, is most relevant, along with **Section 2.5**, **Transport Layer Interface**.

1.3.3 Network Component Implementors

For the implementor of networking software, the Host Requirements RFCs represent the official definition of the requirements for an implementation of the IPS, it is not the intention of this guide to subvert this authority and recommend an alternative standard for implementations. However, this guide presents valuable information regarding the capabilities of current networked UNIX and derivative systems, a vital factor when considering interworking issues. For this group of readers, the whole of the guide contains relevant information, the reader should read those sections relevant to the protocols being addressed.

1.4 DERIVATION OF CURRENT COMMON PRACTICE

This section describes the process used to determine what is ‘‘current common practice’’ in the world of IPS protocols. This process was used to determine which protocols form the base set required by any implementation of the internet protocol suite for UNIX and UNIX-derived systems. Then, for each of the selected protocols, it was used to determine the status of each protocol feature or option.

1.4.1 Definition of Current Common Practice

As explained in **Section 1.2**, **Scope**, it is possible to categorise IPS implementations based upon their ‘‘architecture’’. However, despite the diversity of architectures, the *functional base* for the majority of modern IPS implementations is defined by the capabilities of the TCP/IP implementation provided as part of the 4.3 BSD version of UNIX distributed by the University of California at Berkeley. In addition the 4.3 BSD software has been modified by an update known as ‘‘TAHOE’’ which added significant performance enhancements and fixed a number of bugs. Some suppliers have included some of the features of this update in their products. Older implementations may be based upon the previous release of this code, 4.2 BSD.

Suppliers have enhanced their offerings from this base in a number of ways. Support for alternative link-layer protocols have been added, and some suppliers have integrated more up-to-date versions of the application-level utilities such as file transfer and virtual terminal access. Some measure of integration with other protocol suites has been added at the transport interface and in some of the higher-level protocols such as electronic mail. Additional higher-level application protocols, such as networked file access, have been added.

Thus for the purposes of this guide:

Current common practice is defined by the protocols and protocol options provided by 4.3 BSD. The advanced features provided by 4.3 BSD TAHOE and by public domain versions of the application-level protocols are also sufficiently widely implemented to qualify as common practice. Lastly the guide recognises the significant number of systems which were based upon 4.2 BSD.

1.4.2 Determination of Current Common Practice

In the production of this guide, ‘‘current common practice’’ was determined by examining a sample of IPS products for UNIX and UNIX-derived systems, which are representative of the classes of implementation defined in **Section 1.2**, **Scope**. In addition, at the application layers, the capabilities of public domain versions of the utility programs were examined.

- For the application layers: The representative implementations and public domain software versions were examined in detail using documentation, sources and observation of product behaviour to determine how they match the requirements.
- For the transport layer and below: Here the features of an implementation are less easily determined from documentation or by observation, and common practice was determined largely by examination of the source code for the 4.3 BSD implementation with input from documentation and observation where possible.
- In addition, the release note for 4.3 BSD was used to identify differences from the 4.2 BSD version. The source code and release note for 4.3 BSD TAHOE were used to identify additional functionality added by that update.

When determining current practice, an attempt was made to filter out simple implementation bugs. There are also a number of undocumented configuration options, usually set by patching kernel variables. Such features are noted as such as a supplier may not support the product when an undocumented option has been selected.

Chapter 2

Communications Protocols

2.1 PHYSICAL LAYER

This section addresses interworking issues at the physical layer. The Host Requirements RFCs, which define the scope of much of this guide, do not consider this layer. However, it is felt that there are significant compatibility issues at this layer which need to be highlighted. As this guide primarily addresses hosts connected to Ethernet and 802.3 LANS, that is the only network technology addressed here.

2.1.1 Physical Layer Requirements

This subsection lists specific requirements for Ethernet and IEEE 802.3 networks which highlight potential interworking problems. The Ethernet protocol is defined by the specification ''Ethernet - A Local Area Network'' version 2.0 [PH-ENET2]. There is a version 1.0 of the standard [PH-ENET1] which is now obsolete. The two Ethernet standards are electrically incompatible, using different coupling between controller and transceiver. The IEEE version is defined by ''IEEE Standards for Local Area Networks: Carrier Sense Multiple Access with Collision Detection (CSMA/CD)'' [PH-802.3]. Ethernet 2.0 and IEEE 802.3 are electrically compatible but the Ethernet frame format differs from that defined for IEEE 802.3. The compatibility issues raised by the frame format (encapsulation) are discussed in **Section 2.2**, **Link Layer**. This section restricts itself to considering the physical layer compatibility issues.

The terminology employed here is that of IEEE 802.3. The equivalent Ethernet term is generally given in brackets.

- **MAU/DTE MUST support the Signal Quality Error (SQE) test.**
  Both the Ethernet version 2.0 and the IEEE 802.3 specifications specify a test of the Signal Quality Error message (collision detect) to be performed after a packet has been transmitted. The test requires the MAU (transceiver) to pulse the SQE signal following each transmission. The DTE (controller) must look for this signal after each transmission and if not detected, close itself down and refuse to transmit any further packets. The intention is to provide protection against broken signal wires or faulty MAU causing the station to transmit at the wrong moment.

- **MAU MAY disable SQE test.**
  In order to avoid upsetting Ethernet version 1 controllers which would see the SQE test message as a genuine collision, a MAU (transceiver) should include a method of disabling the feature.

- **DTE SHOULD disable itself when SQE test fails.**
  In order to implement the test correctly a DTE (controller) should refuse to transmit further packets when no SQE pulse is detected.

- **DTE MUST indicate failure when SQE test fails.**
  As a minimum the DTE (controller) must provide the host system with an indication that there may be a fault in the MAU (transceiver) or AUI cable (transceiver cable).

- **AUI fixings SHOULD be the standard slide-latch.**
  This method of fixing is generally reviled as being unreliable. However, failure to use the standard connector may cause connection problems.
- **AUI fixings MAY be non-standard screw-and-post.**
  Some suppliers of MAU/AUI cable/DTEs have chosen to provide this more solid fixing method. However, in a homogeneous hardware environment, non-standard fixings cause significant problems.

2.1.2 Physical Layer Requirements Summary

This subsection presents a summary table in the style of those included in the Host Requirements RFCs. The table summarises the requirements listed in **Section 2.1.1**, **Physical Layer Requirements**.

FEATURE	MUST	SHLD	MAY	SHLD NOT	MUST NOT
SQE test:					
MAU/DTE support SQE test	x				
MAU disable SQE test			x		
DTE disable itself on SQE test fail		x			
DTE indicate SQE test failure	x				
AUI fixings:					
Standard slide latch		x			
Screw-and-post			x		

2.2 LINK LAYER

This section of the guide concerns itself with the mapping of network technologies into the services required by the Internet Protocol (IP). The equivalent section of the Host Requirements RFC [HR-COMMS] restricts itself to the requirements for running IP over Ethernet and IEEE 802.2 LANs. There is a significant level of support for running the IP protocol over serial lines using the SLIP protocol, in the future PPP is likely to replace SLIP as the protocol of choice for serial lines. As discussed in **Section 1.2**, **Scope**, the use of serial line link protocols is considered in **Chapter 4**, **Gateway Systems**.

The subsequent subsections cover the link layer requirements specified by [HR-COMMS], listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Link Layer Requirements Summary Table from the RFC is reproduced, highlighting the points of divergence.

2.2.1 Link Layer Requirements

The requirements are defined in the RFC ''Host Requirements - Communications Layers'' [HR-COMMS], and cover support for the trailers and ARP protocols, Ethernet and IEEE 802.2 encapsulation and the link-layer interface to the IP layer. Associated RFCs are:

- RFC-826 ''Ethernet Address Resolution Protocol'' [LK-ARP];
- RFC-893 ''Trailer Encapsulations'' [LK-TRAIL];
- RFC-894, ''Standard for Transmission of IP Datagrams over Ethernet Networks'' [LK-ENET], and
- RFC-1042 ''Standard for Transmission of IP Datagrams over IEEE 802 Networks'' [LK-802].

In general, all implementations support Ethernet encapsulation and the ARP and trailer protocols. Support for IEEE 802.2 encapsulation (using the LLC and SNAP protocols) is included for those hosts which support IEEE 802.5 (Token Ring) adapters. There is little support of, or use for IEEE 802.2/IEEE 802.3 networks, although a number of implementations appear to have the hooks in place.

The new version of the Berkeley distribution, 4.4 BSD, is likely to enhance the interface between the link and internet layers. IEEE 802.2 encapsulation is being added, but will only support OSI protocols.

The remainder of this section consists of a list of requirements which current implementations generally do not meet. Where significant, future support by 4.4 BSD is noted. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **2.3.1 - MUST NOT send trailers without negotiation.**
  Some pre-4.3 BSD implementations do not implement trailer negotiation. Conformant implementations should be aware that some hosts may use trailer encapsulation without being given permission and should tolerate this behaviour.
- **2.3.2.1 - MUST prevent ARP floods.**
  The 4.3 BSD sources do not include a mechanism for limiting the rate of ARP requests for a particular IP address. None of the implementations examined appeared to implement such a mechanism. The 4.4 BSD ARP implementation may be enhanced to

ensure that multiple requests for resolution of the same address result in only one broadcast request.

- **2.3.2.1 - ARP cache timeout SHOULD be configurable.**
  In general the ARP cache timeout is fixed. However, at least one implementation does provide a configurable timeout via the *arp* utility.
- **2.3.3 - SHOULD send/receive RFC-1042 encapsulation.**
  There is little current support for the use of RFC-1042 encapsulation for IEEE 802.3 LANs although several implementations appear to have hooks in place to do so.
- **2.3.3 - MUST NOT send K1=6 encapsulation.**
  There is isolated use of this type of encapsulation in some proprietary applications but it is not used for systems where general interworking is required.
- **2.4 - Link layer MUST report b'casts to IP layer.**
  The 4.3 BSD Link-IP interface does not include an indication of whether the datagram was received in a broadcast packet. 4.4 BSD may enhance the interface between the link and internet layers to allow the link protocol to indicate when a packet has been received in a broadcast packet.
- **2.4 - IP layer MUST pass TOS to link layer.**
  The 4.3 BSD IP-link interface does not pass the TOS value for the packet to the link layer explicitly, only as part of the IP packet. Support for passing the TOS value is unlikely to be added by 4.4 BSD as the link layer can ''discover'' it by looking into the IP packet.

2.2.2 Link Layer Requirements Summary

This subsection reproduces the Link Layer Requirements Summary given in the RFC [HR-COMMS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 2.2.1**, **Link Layer Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 2.2.1**, **Link Layer Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Trailer encapsulation	2.3.1				x		
Send trailers by default without negotiation	2.3.1	2					x
ARP							
Flush out-of-date ARP cache entries	2.3.2.1		x				
Prevent ARP floods	2.3.2.1	1	x				
Cache timeout configurable	2.3.2.1	1		x			
Save at least one (latest) unresolved pkt	2.3.2.2			x			
Ethernet and IEEE 802 encapsulation							
Host able to:							
Send & receive RFC-894 encapsulation	2.3.3		x				
Receive RFC-1042 encapsulation	2.3.3	1		x			
Send RFC-1042 encapsulation	2.3.3	1			x		
Then config. sw. to select, RFC-894 dflt	2.3.3	-	x				
Send K1=6 encapsulation	2.3.3	2					x
Use ARP on Ethernet and IEEE 802 nets	2.3.3		x				
Link layer report b'casts to IP layer	2.4	1	x				
IP layer pass TOS to link layer	2.4	1	x				
No ARP cache entry treated as dest. unreach	2.4						x

2.3 INTERNET LAYER

This layer addresses the requirements at the internet layer of the IPS. This layer is roughly equivalent to the network layer of the OSI seven-layer model. At this level there are two protocols:

- **IP - Internet Protocol.**
  This protocol implements datagram routing and forwarding with fragmentation and reassembly as required by the underlying link layers. The protocol does not provide flow control and error recovery.
- **ICMP - Internet Control Message Protocol.**
  This protocol is functionally a part of IP but uses the services of the IP protocol to generate and receive control and error messages.

The subsequent subsections cover the internet layer requirements specified by [HR-COMMS], listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Internet Layer Requirements Summary Table from the RFC is reproduced, highlighting the points of divergence.

2.3.1 Internet Layer Requirements

The requirements are defined in the RFC ''Host Requirements - Communications'' [HR-COMMS] and cover support for both the IP and ICMP protocols. Associated base RFCs are:

- RFC-791, ''Internet Protocol'' [IL-IP];
- RFC-792, ''Internet Control Message Protocol'' [IL-ICMP], and
- RFC-950, ''Internet Standard Subnetting Procedure'' [IL-SUB].

Generally, IP is fully implemented, including routing, forwarding and packet fragmentation/reassembly. All ICMP messages and IP options are supported, including updating of Source Route, Record Route and Time Stamp options. Validation of source and destination addresses is weak and many of the required ''configuration switches'' are not provided. Multicasting is not yet supported.

The new version of the Berkeley distribution, 4.4 BSD, is likely to comply with most of the host requirements at this layer. In particular this is likely to include the following enhancements: comprehensive address validation, in particular, ''specific addresses'' are to be used more consistently in reply datagrams; and enhanced handling of routing table and gateway entries, mostly by adding functionality to a new daemon process.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Where significant, future support by 4.4 BSD is noted. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section. As discussed in **Section 1.4**, **Derivation of Current Common Practice**, the functional reference for these implementations is the networking software which formed part of the Berkeley 4.3 BSD distribution. Where a requirement is supported other than in the ''pre-4.3'' implementations it is indicated as such in the Requirements Summary table and it is not included here.

- **3.1 - MUST provide configuration switch for embedded gateway code.**
  Datagrams are forwarded by default when the host has more than one interface configured. It is usually possible to disable datagram forwarding but the mechanism for doing so involves applying a patch to the operating system kernel. To be truly configurable it should be possible to control forwarding via the standard system configuration process.
- **3.1 - SHOULD be able to log discarded datagrams.**
  No provisions are made for logging the contents of discarded datagrams. All implementations count discards.
- **3.2.1.1 - MUST silently discard Version != 4.**
  No check is made upon the version number of IP datagrams received.
- **3.2.1.3 - Src address MUST be host's own IP address.**
  The source address is restricted to match the specific address or broadcast address of one of the local interfaces.
- **3.2.1.3 - MUST silently discard datagram with bad dest addr.**
  As a consequence of the automatic gateway code, a host will try to forward the datagram if it has more that one interface and a route for the destination. If the datagram cannot be forwarded, the host may reply with an ICMP ''Dest Unreach'' message or discard silently.
- **3.2.1.3 - MUST silently discard datagram with bad source address.**
  No validation is performed on the source address in a received datagram.
- **3.2.1.5 - MAY retain same ID field in identical datagram.**
  Re-use of IP datagrams is not supported.
- **3.2.1.7 - Fixed TTL MUST be configurable.**
  In most systems the TTL value is ''**#define**d'' and is not configurable. However, in some systems it may be defined as a kernel variable and thus be configurable. For 4.4 BSD, this value may be configurable.
- **3.2.1.8c - MUST build correct (non-redundant) return route.**
  No validation of a received source route is performed. If the route incorrectly contains the source host as the first hop then the reversed route will include it as the final hop. For 4.4 BSD, a received source route may be validated and redundant hops removed.
- **3.2.1.8c - MUST NOT send multiple SR options in one header.**
  No validation of the transmitted options is performed other than the extraction of the first-hop gateway from a source route option.
- **3.2.2 - (ICMP error) Included octets MUST be same as received.**
  This requirement is met in spirit. However, The IP header in an ICMP error reply has the *ip_len* value re-calculated to reflect the length of the included datagram, the *ip_off* field is returned in host-byte order which may differ from the network order.
- **3.2.2 - MUST NOT send ICMP error for IP b'cast/m'cast.**
  This is supported in principle but certain types of broadcast are not filtered out. In particular, ''limited broadcasts'' (-1) and ''network directed broadcasts'' on a subnetted network (<*net*>,-1,-1) are not recognised.

- **3.2.2 - MUST NOT send ICMP error for link-layer b'cast.**
  The IP layer is unaware of what sort of link-layer addressing is used and thus cannot meet this requirement. For 4.4 BSD, the IP-link layer interface may include a ''broadcast'' flag, ICMP will then not send errors for unicast IP packets which arrive in link- layer broadcast packets.
- **3.2.2 - MUST NOT send ICMP error to non-unique src address.**
  No validation of source address performed prior to building error message.
- **3.2.2.1 - MUST generate dest. unreachable (code 2/3).**
  UNREACH_PORT is generated but UNREACH_PROTO is not.
- **3.2.2.2 - SHOULD NOT send Redirect.**
  By default, a host with more than one interface operates as a gateway, including the sending of Redirect messages. This may be affected by the setting of a gateway configuration switch where provided.
- **3.2.2.2 - MUST handle both host and net redirects.**
  Both type of redirect are handled, however, contrary to the requirements, a REDIRECT_NET can result in the update of a routing table ''Net'' entry.
- **3.2.2.3 - MAY send Source Quench if buffering exceeded.**
  Source quench is only sent when shortage of buffers occurs during fragmentation of a forwarded datagram. No mechanism is provided for generating source quench when buffer shortage affects datagram reception.
- **3.2.2.6 - MAY discard echo to b'cast/m'cast address.**
  Destination address not checked for ICMP request replies.
- **3.2.2.6 MUST use specific-dest addr as echo reply src.**
  This requirement is met in spirit. However the same comments regarding the recognition of broadcast addresses which applied to ICMP error messages apply here.
- **3.2.2.6 - Echo reply SHOULD reflect RR and TS options.**
  These options are stripped out of echo replies.
- **3.2.2.7 - SHOULD NOT send Information Request/Reply messages.** Information request messages ARE ''reflected''.
- **3.2.2.8 - MAY silently discard b'cast/m'cast time stamp.**
  Destination address not checked for ICMP time stamp replies.
- **3.2.2.8 - MUST use specific-dest addr as TS reply src.**
  This requirement is met in spirit. However the same comments regarding the recognition of broadcast addresses which applied to ICMP error messages apply here.
- **3.2.2.8 - TS reply SHOULD reflect RR and TS options.**
  These options are stripped out of time stamp replies.
- **3.2.2.9 - Address mask source MUST be configurable.**
  Address mask determination is always static, via the *ifconfig* utility.
- **3.2.2.9 - SHOULD perform reasonableness check on addr mask.**
  The *ifconfig* utility checks that the subnet mask is consistent with the network mask implied by the internet address. No check is made for a mask of all 1s.

- **3.2.2.9 - MUST NOT send unauthorised addr mask reply msgs.**
  An address mask request always prompts a reply, there is no concept of an authoritative server.
- **3.3.1.2 - SHOULD treat host and net redirect the same.**
  A REDIRECT_NET can result in the update of a routing table ''Net'' entry.
- **3.3.1.2 - MUST support multiple default gateways.**
  Only a single default gateway is supported. For 4.4 BSD it is planned to support this requirement using functionality in a user-level routing daemon process.
- **3.3.1.2 - MAY provide flag: route overridable by redirects.**
  A REDIRECT_NET message may update any route other than the default route. For 4.4 BSD it is planned to support this requirement using functionality in a user-level routing daemon process.
- **3.3.1.3 - SHOULD include TOS in route cache.**
  No TOS field included in cache table.
- **3.3.1.4 - MUST detect failure of next-hop gateway.**
  DYNAMIC route table entries are deleted upon negative advice from the TCP layer, supplied when a threshold for consecutive lost packets is reached. No positive advice or active checking is provided for next-hop gateways. For 4.4 BSD it is planned to support this requirement using functionality in a user-level routing daemon process.
- **3.3.3 - SHOULD send max 576 to off-net destination.**
  In general, implementations do not limit MTU size for remote destinations. However, at least one implementation allows a fixed ''remote MTU'' to be configured for each interface. For 4.4 BSD, the size of off-net packets should be configurable.
- **3.3.3 - MTU of each interface MUST be configurable.**
  In general, MTUs are determined automatically based upon on the underlying network for each interface installed.
- **3.3.3 - MAY provide all-subnets-MTU configuration flag.**
  No all-subnets-MTU flag is provided.
- **3.3.4.2 - SHOULD reply with same addr as spec-dest addr.**
  An attempt is made to detect broadcast destination address and use the equivalent specific destination address. However the same comments regarding the recognition of broadcast addresses which applied to ICMP Error Messages apply here.
- **3.3.4.2 - MAY silently discard d'gram in ''wrong'' interface.**
  Datagram is accepted as long as its destination address matches the specific or broadcast addresses of one of the installed interfaces.
- **3.3.4.2 - MAY only send d'gram through ''right'' interface.**
  No validation of source address is performed (other than checks for INADDR_ANY).
- **3.3.5 - MUST update TTL by gateway rules.**
  Implementations based upon the original 4.3 BSD code may forward datagrams with a TTL of zero, only discarding when a TTL of zero is received. For later versions of BSD (in particular 4.3 BSD TAHOE) this bug has been fixed and datagrams received with a

TTL of one are not forwarded. Implementations which include this fix meet the requirement.

- **3.3.5 - MUST provide configurable switch for non-local SRing.**
  No Source Route forwarding configuration switch is provided.
- **3.2.1.3 - MUST NOT send broadcast addr as IP source addr.**
  No validation of source address is performed at the IP layer (other than checks for INADDR_ANY).
- **3.3.6 - SHOULD silently discard link-layer-only b'cast dgs.**
  The IP layer is unaware of what sort of link-layer addressing is used and thus cannot meet this requirement.
- **3.3.6 - SHOULD default to -1 broadcast.**
  In general, systems based upon the 4.3 BSD code or its variants correctly default the broadcast address. However, at least one product appears to have chosen to the keep ''0'' as the default, presumably to retain compatibility with their installed base.
- **3.3.6 - SHOULD use limited broadcast addr for connected net.**
  Limited broadcast not used, hosts send (subnet) directed broadcasts.
- **3.3.7 - SHOULD support local IP multicasting (RFC-1112).**
  Local multicasting not supported.
- **3.3.7 - MAY support IGMP (RFC-1112).**
  IGMP not supported.
- **3.4 - MUST pass interface ident up to transport layer.**
  The transport layer is not informed which interface a datagram was received on. For 4.4 BSD, the higher layers may be informed of the receiving interface.

2.3.2 Internet Layer Requirements Summary

This subsection reproduces the Internet Layer Requirements Summary given in the RFC [HR-COMMS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 2.3.1**, **Internet Layer Requirements** for possible clarification.

2 Requirement supported with some qualification, see **Section 2.3.1**, **Internet Layer Requirements** for clarification.

3 Requirement not supported by ''pre-4.3'' implementations, see introduction to **Section 2.3.1**, **Internet Layer Requirements** for clarification.

G Gateway requirements are covered in **Chapter 4**, **Gateway Systems**.

\- Requirement not applicable, see a preceding list item.

An entry in the ''FEATURE'' column marked with a double asterisk ''**' represents an explicit requirement in the text of [HR-COMMS] which has been missed from the appropriate summary table, it is included here for completeness.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Implement IP and ICMP	3.1		x				
Handle remote multihoming in application layer	3.1		x				
Support local multihoming	3.1				x		
Meet gateway specs if forward datagrams	3.1		x				
Configuration switch for embedded gateway	3.1	2	x				
Config switch default to non-gateway	3.1	-	x				
Auto-config based on number of interfaces	3.1	-					x
Able to log discarded datagrams	3.1	1		x			
Record in counter	3.1			x			
Silently discard Version != 4	3.2.1.1	1	x				
Verify IP checksum, silently discard d'gram	3.2.1.2		x				
Addressing:							
Subnet addressing (RFC-950)	3.2.1.3	3	x				
Src address must be host's own IP address	3.2.1.3	2	x				
Silently discard datagram with bad dest addr	3.2.1.3	2	x				
Silently discard datagram with bad src addr	3.2.1.3	1	x				
Support reassembly	3.2.1.4		x				
Retain same Id field in identical datagram	3.2.1.5	1			x		
TOS:							
Allow transport layer to set TOS	3.2.1.6		x				
Pass received TOS up to transport layer	3.2.1.6			x			
Use RFC-795 link-layer mappings for TOS	3.2.1.6					x	
TTL:							
Send packet with TTL of 0	3.2.1.7						x
Discard received packets with TTL < 2	3.2.1.7						x
Allow transport layer to set TTL	3.2.1.7		x				
Fixed TTL is configurable	3.2.1.7	1	x				
IP Options:							
Allow transport layer to send IP options	3.2.1.8		x				
Pass all IP options rcvd to higher layer	3.2.1.8		x				
IP layer silently ignore unknown options	3.2.1.8		x				
Security option	3.2.1.8a				x		
Send Stream Identifier option	3.2.1.8b					x	
Silently ignore Stream Identifier option	3.2.1.8b		x				
Record Route option	3.2.1.8d				x		
Timestamp option	3.2.1.8e				x		
Source Route Option:							
Originate & terminate Source Route options	3.2.1.8c		x				
Datagram with completed SR passed to TL	3.2.1.8c		x				
Build correct (non-redundant) return route	3.2.1.8c	1	x				
Send multiple SR options in one header	3.2.1.8c	1					x
ICMP:							
Silently discard ICMP msg with unknown type	3.2.2		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Include more than 8 octets of orig datagram	3.2.2				x		
Included octets same as received	3.2.2	2	x				
Demux ICMP Error to transport protocol	3.2.2		x				
Send ICMP error message with TOS=0	3.2.2			x			
Send ICMP error message for:							
- ICMP error msg	3.2.2						x
- IP b'cast or IP m'cast	3.2.2	2					x
- Link layer b'cast	3.2.2	1					x
- Non-initial fragment	3.2.2						x
- Datagram with non-unique src address	3.2.2	1					x
Return ICMP error msgs (when not prohibited)	3.3.8		x				
Dest Unreachable:							
Generate Dest Unreachable (Code 2/3)	3.2.2.1	2		x			
Pass ICMP Dest Unreachable to higher layer	3.2.2.1		x				
Higher layer act on Dest Unreach	3.2.2.1			x			
Interpret Dest Unreach only as hint	3.2.2.1		x				
Redirect:							
Host send Redirect	3.2.2.2	2				x	
Update route cache when recv Redirect	3.2.2.2		x				
Handle both Host and Net Redirects	3.2.2.2	2	x				
Discard illegal Redirect	3.2.2.2			x			
Source Quench:							
Send Source Quench if buffering exceeded	3.2.2.3	2			x		
Pass Source Quench to higher layer	3.2.2.3		x				
Higher layer act on Source Quench	3.2.2.3			x			
Time Exceeded: pass to higher layer	3.2.2.4		x				
Parameter Problem:							
Send Parameter Problem messages	3.2.2.5			x			
Pass Parameter Problem to higher layer	3.2.2.5		x				
Report Parameter Problem to user	3.2.2.5				x		
ICMP Echo Request or Reply:							
Echo server	3.2.2.6		x				
Echo client	3.2.2.6			x			
Discard Echo Request to b'cast address	3.2.2.6	1			x		
Discard Echo Request to m'cast address	3.2.2.6	-			x		
Use specific-dest addr as Echo Reply src	3.2.2.6	2	x				
Send same data in Echo Reply	3.2.2.6		x				
Pass Echo Reply to higher layer	3.2.2.6		x				
Reflect Record Route, Time Stamp options	3.2.2.6	1		x			
Reverse and reflect Source Route option	3.2.2.6		x				
ICMP Information Request or Reply	3.2.2.7	1				x	
ICMP Timestamp and Timestamp Reply:							
Implement request and reply	3.2.2.8				x		
Minimize delay variability	3.2.2.8			x			
Silently discard b'cast Timestamp	3.2.2.8	1			x		
Silently discard m'cast Timestamp src	3.2.2.8	-			x		

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Use specific-dest addr as TS Reply src	3.2.2.8	2	x				
Reflect Record Route, Time Stamp options	3.2.2.8	1		x			
Reverse and reflect Source Route option	3.2.2.8		x				
Pass Timestamp Reply to higher layer	3.2.2.8		x				
Obey rules for ''standard value	3.2.2.8		x				
ICMP Address Mask Request and Reply:							
Addr Mask source configurable	3.2.2.9	1	x				
Support static configuration of addr mask	3.2.2.9		x				
Get addr mask dynamically during booting	3.2.2.9	-			x		
Get addr via ICMP Addr Mask Request/Reply	3.2.2.9	-			x		
Retransmit Addr Mask Req if no Reply	3.2.2.9	-	x				
Assume default mask if no Reply	3.2.2.9	-		x			
Update mask from first Reply only	3.2.2.9	-	x				
Reasonableness check on Addr Mask	3.2.2.9	2		x			
Send unauthorized Addr Mask Reply msgs	3.2.2.9	1					x
Explicitly configured to be an agent	3.2.2.9	-	x				
Static config=> Addr-Mask-Authoritative	3.2.2.9	-		x			
Broadcast Addr Mask Reply when init.	3.2.2.9	-	x				
Routing Outbound Datagrams:							
Use address mask in local/remote decision	3.3.1.1		x				
Operate with no gateways on conn network	3.3.1.1		x				
Maintain ''route cache'' of next-hop gateways	3.3.1.2		x				
Treat Host and Net Redirect the same	3.3.1.2	1		x			
If no cache entry, use default gateway	3.3.1.2		x				
Support multiple default gateways	3.3.1.2	1	x				
Provide table of static routes	3.3.1.2				x		
Flag: route overridable by Redirects	3.3.1.2	1			x		
Key route cache on host, not net address	3.3.1.3				x		
Include TOS in route cache	3.3.1.3	1		x			
Able to detect failure of next-hop gateways	3.3.1.4	2	x				
Assume route is good forever	3.3.1.4	-				x	
Ping gateways continuously	3.3.1.4	-					x
Ping only when traffic being sent	3.3.1.4	-	x				
Ping only when no positive indication	3.3.1.4	-	x				
Higher and lower layers give advice	3.3.1.4			x			
Switch from failed default g'way to another	3.3.1.5	1	-				
Manual method of entering config info	3.3.1.6		x				
Reassembly and Fragmentation:							
Able to reassemble incoming datagrams	3.3.2		x				
At least 576 byte datagrams	3.3.2		x				
EMTU_R configurable or indefinite	3.3.2			x			
Transport layer able to learn MMS_R	3.3.2		x				
Send ICMP Time Exceeded on reass. timeout	3.3.2		x				
Fixed reassembly timeout value	3.3.2			x			
Pass MMS_S to higher layers	3.3.3		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
MTU of each i/f MUST be configurable **	3.3.3	1	x				
Local fragmentation of outgoing packets	3.3.3				x		
Else don't send bigger than MMS_S	3.3.3		x				
Send max 576 to off-net destination	3.3.3	1		x			
All-Subnets-MTU configuration flag	3.3.3	1			x		
Multihoming:							
Reply with same addr as spec-dest addr	3.3.4.2	2		x			
Allow application to choose local IP addr	3.3.4.2		x				
Silently discard d'gram in ''wrong'' interface	3.3.4.2	1			x		
Only send d'gram through ''right'' interface	3.3.4.2	1			x		
Source-Route forwarding:							
Forward datagram with Source Route option	3.3.5				x		
Obey corresponding gateway rules	3.3.5	G	x				
Update TTL by gateway rules	3.3.5	2	x				
Able to generate ICMP err code 4,5	3.3.5		x				
IP src addr not local host	3.3.5				x		
Update Timestamp, Record Route options	3.3.5		x				
Configurable switch for non-local SRing	3.3.5	1	x				
Defaults to OFF	3.3.5	-	x				
Satisfy gwy access rules for non-local SR	3.3.5	G	x				
If not forward, send Dest Unreach (cd 5)	3.3.5			x			
Broadcast:							
Broadcast addr as IP source addr	3.2.1.3	1					x
Receive 0 or -1 broadcast formats OK	3.3.6	3		x			
Config'ble option to send 0 or -1 b'cast	3.3.6	3			x		
Default to -1 broadcast	3.3.6	2		x			
Recognize all broadcast address formats	3.3.6		x				
Use IP b'cast addr in link-layer b'cast	3.3.6		x				
Silently discard link-layer-only b'cast dgs	3.3.6	1		x			
Use limited Broadcast addr for connected net	3.3.6	1		x			
Multicast:							
Support local IP multicasting (RFC-1112)	3.3.7	1		x			
Support IGMP (RFC-1112)	3.3.7	1			x		
Join all-hosts group at startup	3.3.7	-		x			
Higher layers learn i'face m'cast capability	3.3.7	-		x			
Interface:							
Allow transport to use all IP mechanisms	3.4		x				
Pass interface ident up to transport layer	3.4	1	x				
Pass all IP options up to transport layer	3.4		x				
Transport can send certain ICMP messages	3.4		x				
Pass spec'd ICMP messages up to transport	3.4		x				
Include IP hdr+8 octets or more from orig.	3.4		x				

2.4 TRANSPORT LAYER

Two protocols are provided at the transport layer:

- **TCP - Transmission Control Protocol.**
  This is the connection-orientated protocol providing a reliable sequenced full-duplex communications channel.
- **UDP - User Datagram Protocol.**
  The connectionless protocol for sending and receiving datagrams. Datagrams are un-acknowledged and neither delivery nor sequencing are guaranteed.

The subsequent subsections cover the transport layer requirements specified by [HR-COMMS] for each of these protocols, listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The TCP and UDP Requirements Summary tables from the RFCs are reproduced, highlighting the points of divergence.

2.4.1 TCP Requirements

The requirements are defined in the RFC ''Host Requirements - Communications Layers'' [HR-COMMS]. The base RFC is: RFC-793, ''Transmission Control Protocol'' [TL-TCP], and there are a number of associated RFCs defining options and clarifying the protocol and its implementation.

Generally the TCP protocol is fully implemented, including the MSS option. Both sender and receiver ''silly window syndrome'' algorithms are supported. Implementations of the basic 4.3 BSD code provide only the ''standard'' round trip estimation algorithms and window management. More modern products should include the performance enhancements introduced by the TAHOE version of 4.3 BSD, including the ''slow-start'' and ''congestion-avoidance'' algorithms. There are weaknesses in the validation of source and destination addresses and no interface is provided to TTL and TOS parameters. No interface is provided for the indication of ''soft errors'' (i.e. Network Unreachable or Source Route failure).

The new version of the Berkeley distribution, 4.4 BSD, is likely to comply with most of the host requirements at this layer. In particular this is likely to include the following enhancements: comprehensive address validation; an interface for setting TOS and TTL values in datagrams; an interface for reporting ''soft errors'' (such as routing changes) to TCP users; and system-wide keep-alive interval configuration.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Where significant, future support by 4.4 BSD is noted. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section. As discussed in **Section 1.4**, **Derivation of Current Common Practice**, the functional reference for these implementations is the networking software which formed part of the Berkeley 4.3 BSD distribution. Where a requirement is supported other than in the ''pre-4.3'' implementations it is indicated as such in the Requirements Summary table and it is not included here.

- **4.2.2.2 - MAY aggregate or queue un-pushed data.**
  The interfaces provided for applications do not include a PUSH flag. Data transfer is

entirely under the control of the TCP. On receipt, data is made available to the application immediately, unpushed data is not buffered.

- **4.2.2.2 - SEND call MAY specify PUSH.**
  Neither the Sockets, TLI or XTI interfaces include a push flag in the send call.
- **4.2.2.2 - MAY notify receiving ALP of PSH.**
  Neither the Sockets, TLI or XTI interfaces include a push indicator in the RECEIVE call.
- **4.2.2.3 - SHOULD handle window as 32-bit number.**
  The window size is handled as an unsigned 16-bit number.
- **4.2.2.4 - Urgent pointer MUST point to last octet.**
  The urgent pointer points to the byte following the last octet of urgent data.
- **4.2.3.10 - MUST NOT OPEN to b'cast/m'cast IP address.**
  No filtering of destination address is performed other than mapping INADDR_ANY to a specific local address and mapping INADDR_BROADCAST to a local broadcast address.
- **4.2.3.10 - MUST silently discard seg to b'cast/m'cast address.**
  There is no requirement in the RFC text to match this entry in the summary table. The 4.3 BSD sources only check for a broadcast destination address when in the LISTEN state, no destination address validation is performed for ESTABLISHED connections.
- **4.2.2.13 - SHOULD send RST to indicate data lost on h-duplex cls.**
  A full duplex close is supported: *t_sndrel*( ) and *t_rcvrel*( ) for TLI / XTI; *shutdown*( ) for sockets. Data received by TCP but not yet read by the user is flushed silently. Data received after user close provokes a RST as required. For 4.4 BSD, flushed data should correctly generate a RST to indicate loss of ACKed data.
- **4.2.2.13 - MAY accept SYN from TIME-WAIT state.**
  SYN not accepted in TIME-WAIT state.
- **4.2.2.15 - MUST implement Jacobson slow start algorithm.**
  Not supported by implementations of the basic 4.3 BSD code, more modern products which include the TAHOE performance enhancements do support this algorithm.
- **4.2.2.15 - MUST implement Jacobson congestion-avoidance algorithm.**
  Not supported by implementations of the basic 4.3 BSD code, more modern products which include the TAHOE performance enhancements do support this algorithm.
- **4.2.2.15 - MAY retransmit with same IP ident.**
  New IP packet built on retransmission.
- **4.2.3.1 - MUST implement Karn's algorithm.**
  Not supported by implementations of the basic 4.3 BSD code, more modern products which include the TAHOE performance enhancements do support this algorithm.
- **4.2.3.1 - MUST implement Jacobson's RTO estimation algorithm.**
  Not supported by implementations of the basic 4.3 BSD code, more modern products which include the TAHOE performance enhancements do support this algorithm.
- **4.2.2.17 - MUST implement exponential back-off of retx.**
  More modern products which include the TAHOE performance enhancements do support exponential back-off. The back-off algorithm pre-TAHOE uses an non-

exponential ascending sequence of factors which are applied to the base retransmit interval.

- **4.2.3.2 - MUST ACK every 2nd full-sized segment.**
  Delayed ACKs are triggered only on a timeout.
  4.4 BSD may implement the enhanced delayed ACK processing.

- **4.2.2.19 - TTL MUST be configurable.**
  TCP's TTL is generally fixed. The 4.3 BSD source sets it to 30 using a **#define**, individual products may have changed this value to reflect the growth in the ''diameter'' of the Internet. For 4.4 BSD the TTL value may be configurable by the TCP user.

- **4.2.3.5 - MUST allow ALP to set R2 threshold.**
  The R2 (disconnect) threshold is not configurable. 4.4 BSD may allow this threshold to be configured.

- **4.2.3.5 - SHOULD inform ALP of R1<=retxs<R2.**
  No mechanism exists for informing application when retries exceed the R1 threshold (re-route trigger). For 4.4 BSD, a mechanism may be included for informing applications about ''soft errors'' like the retry threshold.

- **4.2.3.6 - Keep-alive interval MUST be configurable.**
  For implementations of the basic 4.3 BSD code, the timeout interval is not configurable. Implementations which include the TAHOE enhancements allow the interval to be configured on a system-wide basis by means of a patch to the system kernel, configuration per connection is not provided. For 4.4 BSD, this interval may be configurable on a system-wide basis.

- **4.2.3.6 - Default keep-alive interval MUST be at least 2 hrs.**
  The default timeout interval is fixed at 6 minutes.

- **4.2.3.8 - MAY support RR/TS option.**
  Any IP options may be set for output, all incoming IP options are discarded apart from source route.

- **4.2.3.8 - Later source route SHOULD override current.**
  Incoming source route only accepted from SYN packets. Later SRs are ignored.

- **4.2.3.9 - Dest. unreach. (0,1,5) SHOULD inform ALP.**
  The application is not informed. The appropriate error code is passed to the socket of each interested application but the applications are not woken-up to receive it. For 4.4 BSD, the soft-error reporting mechanism may allow the application to be informed of such responses.

- **4.2.3.9 - Dest. unreach. (2-4) SHOULD abort connection.**
  The connection is not aborted. These codes are not distinguished from the other destination unreachable codes.

- **4.2.3.9 - Source quench SHOULD trigger slow start.**
  Older implementations may not support the slow start algorithm, in this case SQ causes a progressive closing of the transmit window, known as a congestion window. Subsequent ACKs then progressively re-open the window to the advertised value.

- **4.2.3.9 - TE/PP SHOULD inform ALP, don't abort.**
  Comment as for dest. unreach. (0,1,5). For 4.4 BSD, the soft-error reporting mechanism may allow the application to be informed of such responses.
- **4.2.3.10 - MUST reject OPEN call to invalid IP address.**
  No filtering of destination address is performed other than mapping INADDR_ANY or INADDR_BROADCAST to the appropriate IP address of the local primary interface.
- **4.2.3.10 - MUST reject SYN from invalid IP address.**
  No validation of source address by TCP.
- **4.2.3.10 - MUST discard SYN to invalid IP address.**
  The LISTEN state silently discards segments with a broadcast destination address, however Limited Broadcast address (-1) is not recognised.
- **4.2.4.1 - MUST provide error report mechanism.**
  No mechanism is provided for reporting soft errors to the application. For 4.4 BSD it is likely that an interface will be provided for reporting soft errors.
- **4.2.4.2 - MUST allow ALP to specify TOS for sending.**
  No interface is provided for setting the TOS field in outgoing packets.
  An interface may be provided under 4.4 BSD.
- **4.2.4.2 - MAY pass received TOS up to ALP.**
  No interface is provided for passing the TOS field from incoming packets upwards.
  An interface may be provided under 4.4 BSD.
- **4.2.4.3 - MAY provide FLUSH call.**
  No mechanism for discarding unsent data.

2.4.2 TCP Requirements Summary

This subsection reproduces the TCP Requirements Summary given in the RFC [HR-COMMS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 2.4.1**, **TCP Requirements** for possible clarification.

2 Requirement supported with some qualification, see **Section 2.4.1**, **TCP Requirements** for clarification.

3 Requirement not supported by ''pre-4.3'' implementations, see the introduction to **Section 2.4.1**, **TCP Requirements** for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Aggregate or queue un-pushed data	4.2.2.2	1			x		
Sender collapse successive PSH flags	4.2.2.2	-		x			
SEND call can specify PUSH	4.2.2.2	1			x		
If cannot: sender buffer indefinitely	4.2.2.2						x
If cannot: PSH last segment	4.2.2.2		x				
Notify receiving ALP of PSH	4.2.2.2	1			x		
Send max size segment when possible	4.2.2.2			x			
Window:							
Treat as unsigned number	4.2.2.3		x				
Handle as 32-bit number	4.2.2.3	1		x			
Shrink window from right	4.2.2.16					x	
Robust against shrinking window	4.2.2.16		x				
Receiver's window closed indefinitely	4.2.2.17				x		
Sender probe zero window	4.2.2.17		x				
First probe after RTO	4.2.2.17			x			
Exponential back-off	4.2.2.17			x			
Allow window stay zero indefinitely	4.2.2.17		x				
Sender timeout OK conn with zero wind	4.2.2.17						x
Urgent Data:							
Pointer points to last octet	4.2.2.4	1	x				
Arbitrary length urgent data sequence	4.2.2.4		x				
Inform ALP asynchronously of urgent data	4.2.2.4		x				
ALP can learn if/how much urgent data Q'd	4.2.2.4		x				
TCP Options:							
Receive TCP option in any segment	4.2.2.5		x				
Ignore unsupported options	4.2.2.5		x				
Cope with illegal option length	4.2.2.5		x				
Implement sending & receiving MSS option	4.2.2.6		x				
Send MSS option unless 536	4.2.2.6			x			
Send MSS option always	4.2.2.6				x		
Send-MSS default is 536	4.2.2.6		x				
Calculate effective send seg size	4.2.2.6		x				
TCP Checksums:							
Sender compute checksum	4.2.2.7		x				
Receiver check checksum	4.2.2.7		x				
Use clock-driven ISN selection	4.2.2.9		x				
Opening Connections:							
Support simultaneous open attempts	4.2.2.10		x				
SYN-RCVD remembers last state	4.2.2.11		x				
Passive Open call interfere with others	4.2.2.18						x
Function: simultan. LISTEN for same port	4.2.2.18		x				
Ask IP for src address for SYN if necc.	4.2.3.7		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Otherwise, use local addr of conn.	4.2.3.7		x				
OPEN to broadcast / multicast IP address	4.2.3.10	1					x
Silently discard seg to b'cast / m'cast addr	4.2.3.10	1	x				
Closing Connections:							
RST can contain data	4.2.2.12			x			
Inform application of aborted conn	4.2.2.13		x				
Half-duplex close connections	4.2.2.13	2			x		
Send RST to indicate data lost	4.2.2.13	1		x			
In TIME-WAIT state for 2xMSL seconds	4.2.2.13		x				
Accept SYN from TIME-WAIT state	4.2.2.13	1			x		
Retransmissions:							
Jacobson slow start algorithm	4.2.2.15	2	x				
Jacobson congestion-avoidance algorithm	4.2.2.15	2	x				
Retransmit with same IP ident	4.2.2.15	1			x		
Karn's algorithm	4.2.3.1	2	x				
Jacobson's RTO estimation alg.	4.2.3.1	2	x				
Exponential back-off	4.2.3.1	2	x				
SYN RTO calc same as data	4.2.3.1			x			
Recommended initial values and bounds	4.2.3.1			x			
Generating ACKs:							
Queue out-of-order segments	4.2.2.20			x			
Process all Q'd before send ACK	4.2.2.20		x				
Send ACK for out-of-order segment	4.2.2.21				x		
Delayed ACKs	4.2.3.2			x			
Delay < 0.5 seconds	4.2.3.2		x				
Every 2nd full-sized segment ACKed	4.2.3.2	1	x				
Receiver SWS-avoidance algorithm	4.2.3.3		x				
Sending data:							
Configurable TTL	4.2.2.19	1	x				
Sender SWS-avoidance algorithm	4.2.3.4		x				
Nagle algorithm	4.2.3.4			x			
Application can disable Nagle algorithm	4.2.3.4		x				
Connection failures:							
Negative advice to IP on R1 retxs	4.2.3.5		x				
Close connection on R2 retxs	4.2.3.5		x				
ALP can set R2 threshold	4.2.3.5	1	x				
Inform ALP of R1 <= retxs < R2	4.2.3.5	1		x			
Recommended values for R1, R2	4.2.3.5			x			
Same mechanism for SYNs	4.2.3.5		x				
R2 at least 3 minutes for SYN	4.2.3.5		x				
Keep-alive:							
Send keep-alive packets	4.2.3.6				x		
- Application can request	4.2.3.6		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
- Default is ‘‘off’’	4.2.3.6		x				
- Only send if idle for interval	4.2.3.6		x				
- Interval configurable	4.2.3.6	2	x				
- Default at least 2 hrs	4.2.3.6	1	x				
- Tolerant of lost ACKs	4.2.3.6		x				
IP Options:							
Ignore options TCP doesn’t understand	4.2.3.8		x				
Time stamp support	4.2.3.8	2			x		
Record route support	4.2.3.8	2			x		
Source route:							
ALP can specify	4.2.3.8		x				
Overrides src rt in datagram	4.2.3.8		x				
Build return route from src rt	4.2.3.8		x				
Later src route overides	4.2.3.8	1		x			
ICMP messages:							
Receive ICMP messages from IP	4.2.3.9		x				
Dest. unreach (0,1,5) => inform ALP	4.2.3.9	1		x			
Dest. unreach (0,1,5) => abort conn	4.2.3.9						x
Dest. unreach (2-4) => abort conn	4.2.3.9	1		x			
Source quench => slow start	4.2.3.9	2		x			
Time exceeded => tell ALP, don’t abort	4.2.3.9	1		x			
Param problem => tell ALP, don’t abort	4.2.3.9	1		x			
Address validation:							
Reject OPEN call to invalid IP address	4.2.3.10	1	x				
Reject SYN from invalid IP address	4.2.3.10	1	x				
Silently discard SYN to b’cast / m’cast addr	4.2.3.10	2	x				
TCP / ALP interface services:							
Error report mechanism	4.2.4.1	1	x				
ALP can disable error report routine	4.2.4.1	-		x			
ALP can specify TOS for sending	4.2.4.2	1	x				
Passed unchanged to IP	4.2.4.2	-		x			
ALP can change TOS during connection	4.2.4.2	-		x			
Pass received TOS up to ALP	4.2.4.2	1			x		
FLUSH call	4.2.4.3	1			x		
Optional local IP addr parm. in OPEN	4.2.4.4		x				

2.4.3 UDP Requirements

The requirements are defined in the RFC ‘‘Host Requirements - Communications Layers’’ [HR-COMMS]. The base RFC for UDP is RFC-768 ‘‘User Datagram Protocol’’ [TL-UDP].

The UDP protocol is very simple and implementations are generally complete. There are weaknesses in address validation and no interface is provided to TTL and TOS values.

The new version of the Berkeley distribution, 4.4 BSD, is likely to comply with most of the host requirements at this layer. In particular this is likely to include the following

enhancements: comprehensive address validation; an interface for setting TOS and TTL values in datagrams.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Where significant, future support by 4.4 BSD is noted. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **4.1.3.2 - MUST pass rcv'd IP options to applic layer.**
  The UDP protocol discards all incoming options.
- **4.1.3.3 - MUST pass ICMP messages up to applic layer.**
  All messages apart from redirect (which triggers a re-route) and source quench (which is discarded) are passed upwards and receiving applications are woken-up.
- **4.1.3.4 - MAY provide sender option to not generate checksum.**
  It is usually possible to disable UDP checksum generation but the mechanism for doing some involves applying a patch to the operating system kernel.
- **4.1.3.4 - default MUST be to checksum.**
  For user processes UDP checksums are generated by default; however, some implementations provide an interface to UDP for use by kernel processes such as the NFS protocols. Datagrams sent by this interface are sent with a zero checksum by default.
- **4.1.3.4 - MAY provide receiver option to require checksum.**
  There is no socket or system level option to discard zero-summed UDP datagrams.
- **4.2.3.5 - MUST pass specific-dest. address to application.**
  No validation/conversion of a datagram's destination address is performed, the address is passed to the application as received. For 4.4 BSD, the ''specific destination'' at which a broadcast datagram was received should be made available to the application.
- **4.1.3.6 - Bad IP src addr MUST be silently discarded by UDP/IP.**
  No validation of received source address by UDP, see appropriate internet layer requirement for IP action.
- **4.1.3.6 - MUST only send valid IP source address.**
  The chosen source address is validated against the addresses of the local interfaces, however the broadcast address of a local interface may be used as the source address for a datagram.
- **4.1.4 - MUST be able to spec TTL, TOS, IP opts when send dg.**
  UDP sends a fixed TTL and a TOS of zero, IP options may be specified via the transport layer interface. For 4.4 BSD, an interface may be provided for the application to set these fields in the IP packet.
- **4.1.4 - MAY pass received TOS up to applic layer.**
  The received TOS is NOT made available to the application. 4.4 BSD may allow applications to access this IP field.

2.4.4 UDP Requirements Summary

This subsection reproduces the UDP Requirements Summary given in the RFC [HR-COMMS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 2.4.3**, **UDP Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 2.4.3**, **UDP Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
UDP send Port Unreachable	4.1.3.1			x			
IP Options in UDP:							
- Pass rcv'd IP options to applic layer	4.1.3.2	1	x				
- Applic layer can specify IP options in Send	4.1.3.2		x				
- UDP passes IP options down to IP layer	4.1.3.2		x				
Pass ICMP msgs up to applic layer	4.1.3.3	2	x				
UDP checksums:							
- Able to generate/check checksum	4.1.3.4		x				
- Silently discard bad checksum	4.1.3.4		x				
- Sender option to not generate checksum	4.1.3.4	2			x		
- Default is to checksum	4.1.3.4	2	x				
- Receiver option to require checksum	4.1.3.4	1			x		
UDP multihoming:							
- Pass spec-dest addr to application	4.1.3.5	1	x				
- Applic layer can specify local IP addr	4.1.3.5		x				
- Applic layer specify wild local IP addr	4.1.3.5		x				
- Applic layer notified of local IP addr used	4.1.3.5			x			
Bad IP src addr silently discarded by UDP/IP	4.1.3.6	1	x				
Only send valid IP source address	4.1.3.6	2	x				
UDP application interface services:							
- Full IP interface of 3.4 for application	4.1.4		x				
- Able to spec TTL, TOS, IP opts when send dg	4.1.4	2	x				
- Pass received TOS up to applic layer	4.1.4	1			x		

2.5 TRANSPORT LAYER INTERFACE

The interface to the services provided by the transport layer is key to production of portable networked application software, both across different IPS implementations and across different underlying transport layer protocols. The host requirements RFCs specify the services this interface must provide, but stop short of sanctioning a particular application programming interface (API). As source-code level portability for applications is a key X/Open objective this guide must go further than the RFCs. There are three current interfaces to the services of the transport layer:

- **Sockets.** Based upon the socket library provided by the Berkeley 4.3 BSD distribution.
- **Transport Layer Interface (TLI).** As defined for UNIX system V, using the STREAMS architecture to provide a transport-provider-independent interface to transport services.
- **XTI**. Proposed by X/Open as the standard for an interface independent of operating system and transport provider. XTI is a non-proprietary version of System V's TLI.

The sockets interface, because it is so widely implemented, currently provides the most portable interface for applications. Even implementations which provide TLI as the native transport interface also provide sockets as a compatibility interface, most application software (including their own Telnet and FTP utilities and daemons) currently uses the socket interface. There is at least one popular IPS implementation based upon an earlier version of the interface which can cause porting problems.

The socket interface has two basic drawbacks: Firstly, there is no formal definition of the interface other than that given by the manual pages for the component functions and source code of the original implementation. Secondly, it falls short of being transport provider independent. This makes it difficult to write applications which are truly portable across protocol suites. The forthcoming 4.4 BSD version includes an enhanced socket interface which provides support for OSI protocols. This new socket interface allows more transport provider independence.

The TLI is an integral part of UNIX system V as of V.3 and is supported by many independent IPS implementation vendors. As mentioned above, most TLI-based IPS implementations provide a sockets interface as a compatibility library. TLI goes further than sockets towards being transport provider independent, however it still fails to insulate the application fully from the details of the transport's address formats.

Its primary disadvantage is that it is currently less widely implemented than the socket interface.

The XTI interface, which is based upon the TLI, is the standard transport provider interface for X/Open compliant systems and is expected to be widely adopted. This guide recommends that new IPS implementations implement at least the mandatory XTI functions, and that new applications are implemented to the XTI standard, using the mandatory functions only. The socket interface is still required for backward compatibility.

Chapter 3

Applications and Support

3.1 REMOTE LOGIN

Two protocols are in general use for remote login in IPS implementations:

- **Telnet**
  This protocol, using a reliable transport-layer service (usually TCP), provides a full-duplex communications link for connecting a terminal to a host computer. The protocol provides a ''network virtual Terminal'' with basic capabilities plus an option protocol for negotiating more sophisticated modes of operation.
- **rlogin - Remote login utility.**
  This utility originated as part of the BSD implementation of TCP/IP. It has since been ported to most UNIX derivative systems and even some unrelated systems. It provides the basic full duplex communications path generally required by asynchronous terminals plus some automatic login features for ''trusted hosts''.

As discussed in **Section 1.2**, **Scope**, *rlogin* and the other ''r'' utilities are not covered here due to the lack of any formal specification of the protocols utilised.

The subsequent subsections cover the Telnet protocol, listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Requirements Summary Table from [HR-APPS] is reproduced here with an additional column which highlights the areas of divergence.

3.1.1 Telnet Requirements

For the Telnet protocol, the requirements are specified by the RFC ''Host Requirements - Application and Support'' [HR-APPS] and by the base RFC RFC-854, ''Telnet Protocol Specification'' [RL-TNET] and many associated RFCs that define the options it supports.

In general, the more recent implementations of Telnet, for which the 4.3 BSD implementation appears to be the functional reference, are fairly complete, providing all the control functions and most of the required options. In particular the following options are generally implemented:

- Binary Transmission;
- Echo, and
- Suppress Go-Ahead.

The following options have some level of support:

- **Terminal-type.** This option is widely supported by Server-Telnets but generally only in a primitive form which does not allow server/client negotiation. Client-Telnets do not support the option.
- **Suppress Go-Ahead.** Although both Client- and Server-Telnets will negotiate into and out of Suppress-Go-Ahead mode, half-duplex working is not supported, Go-Ahead commands are always ignored and never sent.

- **Status.** There is isolated support for this option.
- **Extended Option List.** There is isolated support for this option.
- **Timing-Mark.** This option is generally NOT supported, however, a User-Telnet may send a ''DO TIMING-MARK'' option just to provoke a response (the ''WONT TIMING-MARK'' response at least indicates the status of the data on the Telnet link even if the server host has input and output data buffered).

Options specified in recent years, such as Line-Mode, X-Display and Window Size, have not yet been widely implemented (if at all).

The requirements aimed at bringing order to end-of-line processing are supported with some exceptions, in particular Client-Telnets generally send CR NUL as the end-of-line ''character''.

Pre-4.3 BSD implementations are less complete and in particular do not implement the full set of control functions and support fewer options.

More advanced versions of the BSD Telnet source code are now available, giving an indication of the capabilities of the version to be included in the 4.4 BSD distribution. Among possible enhancements are: support for the Line Mode (RFC-1116), X-Display Location (RFC-1096), Window Size (RFC-1073), Terminal Speed (RFC-1079) and Remote Flow Control (RFC-1080) options; enhanced support for the TERMINAL TYPE (RFC-1091) option; Support the ENVIRON option (currently being defined), used to pass information between the local and remote ''environment''; compile-time selection of Type-of-Service for the underlying transport service (allowing selection of a high-bandwidth or low-delay path).

The remainder of this subsection consists of a list of requirements which current implementations of IPS for UNIX and derivative systems generally do not meet. Each bullet point corresponds to an entry in the Requirements Summary table, the reference given directs the reader to the appropriate requirements RFC section. Where a requirement is supported other than in the ''pre-4.3'' implementations it is indicated as such in the Requirements Summary table and is not included here.

- **3.2.8 - MUST Send official name in Term-Type option.**
  This option is supported by all but some older implementations of Telnet. Where it is implemented, the name sent in the option is that held in the Telnet user's *TERM* environment variable and is thus compatible with conventions for terminal naming (i.e. that of the *termcap* and *termdef* terminal capabilities databases) NOT the Assigned Numbers official name.
- **3.2.8 - MUST accept any name in Term-Type option.**
  Again, all but the older implementations initiate Term-Type option negotiations and accept any terminal name in the response. However, not all implementations actually use the information established by the Term-Type option to set the *TERM* environment variable for the terminal session.
- **3.3.3 - SHOULD implement Echo, Status, EOR and Ext. Opts List.**
  The Echo option is fully supported. There are only isolated implementations of Status and Extended Option List options. The end-of-record option does not appear to be implemented.

- **3.3.3 - SHOULD implement Window-Size option.**
  The Window-Size option is not implemented currently. Future versions of Telnet are likely to support it.
- **3.3.4 - User SHOULD be able to enable/disable init negotiations.**
  User-Telnets generally allow some options to be selected before a connection is initiated. However, this falls short of the requirement.
- **3.2.2 - MAY support EOR, EC, EL, Break.**
  Apart from EOR, all these control functions are generally supported.
- **3.2.4 - User-Telnet MAY send Synch after IP, AO, AYT.**
  All but the pre-4.3 implementations can be configured to send a Synch command after an IP, AO or BRK command. Synch is NOT sent after an AYT command. The requirement text only refers to a Synch being sent after an IP, the summary table adds AO and AYT to the list.
- **3.2.4 - Server-Telnet MAY reply Synch to IP.**
  Not supported.
- **3.2.5 - SHOULD NOT send high-order bit in NVT.**
  All implementations will pass data with high-order bit set without having first negotiated Binary mode. This is not a ''deviation'' as such, the ''intention'' of the requirement is to broaden the definition for NVT to allow an ''8-bit NVT'' to be implemented.
- **3.2.5 - SHOULD negotiate Binary if passing high-order bit to application.**
  Not supported.
- **3.3.1 - EOL at server MUST be same as local end-of-line.**
  This is generally supported, however there are isolated implementations that map CR LF differently.
- **3.3.1 - User-Telnet MUST be able to send CR LF, CR NUL or LF.**
  The 4.3 BSD User-Telnet does not provide full control over EOL processing. Later versions of Telnet support the full repertoire of end-of-line conventions.
- **3.3.1 - ASCII user SHOULD be able to select CR LF/CR NUL.**
  No configuration option is provided in the Berkeley 4.3 BSD implementation. Later versions of Telnet do support a user option.
- **3.3.1 - User-Telnet default mode SHOULD be CR LF.**
  The majority of implementations default to CR NUL, there are, however, Client-Telnets which send CR LF by default.
- **3.4.1 - MUST provide escape character**
  All implementations provide an ''enter command mode'' escape. Doubling of this escape to send it as data is not supported by the base 4.3 BSD User-Telnet. Support for escape doubling was added in a later version of the User-Telnet program so some implementations may allow it. Most implementations provide an interactive ''send escape'' when in Telnet command mode.
- **3.4.1 - MAY provide an escape to enter 8-bit values.**
  This is not supported.

- **3.4.3 - SHOULD report TCP connection errors to user.**
  Only ‘‘hard errors’’ such as ‘‘network unreachable’’ are reported to user. Soft errors such as repeated timeouts causing a re-route attempt are not reported.
- **3.4.5 - SHOULD be able to manually restore output mode.**
  No method of restoring output when a a server fails to respond correctly to a Timing-Mark option has been implemented.

3.1.2 Telnet Requirements Summary

This subsection reproduces the Telnet Requirements Summary table given in the RFC [HR-APPS] with an additional ‘‘XGIPS’’ column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.1.1**, **Telnet Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 3.1.1**, **Telnet Requirements**, for clarification.

3 Requirement not supported by ‘‘pre-4.3’’ implementations, see the introduction to **Section 3.1.1**, **Telnet Requirements**.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Option negotiation	3.2.1		x				
Avoid negotiation loops	3.2.1		x				
Refuse unsupported opts	3.2.1		x				
Negotiation OK any time on conn.	3.2.1			x			
Default to NVT	3.2.1		x				
Send official name in Term-Type opt	3.2.8	1	x				
Accept any name in Term-Type opt	3.2.8	2	x				
Implement Binary, suppress-GA opts	3.3.3		x				
Echo, Status, EOL, Ext-Opt-Lst opts	3.3.3	1		x			
Implement Window-Size opt	3.3.4	1		x			
Server initiate mode negotiations	3.3.4			x			
User enable/disable init neg	3.3.4	1		x			
Go-Aheads							
Non-GA server negotiate SGA opt	3.2.2	3	x				
User or server accept SGA opt	3.2.2		x				
User Telnet ignore GAs	3.2.2				x		
Control functions							
Support SE NOP DM IP AO AYT SB	3.2.3	3	x				
Support EOR EC EL Break	3.2.3	2			x		
Ignore unsupported ctrl functions	3.2.3		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Discard urgent data up to DM	3.2.4		x				
User Telnet ''Synch'' after IP/AO/AYT	3.2.4	3		x			
Server Telnet reply Synch to IP	3.2.4	1			x		
Server Telnet reply Synch to AO	3.2.4	3	x				
User Telnet can flush output on IP	3.2.4	3		x			
Encoding							
Send high-order bit in NVT mode	3.2.5	2				x	
Send high-order bit as parity bit	3.2.5						x
Negot. BIN if pass high-ord. bit	3.2.5	1		x			
Always double IAC data byte	3.2.6		x				
Double IAC data byte in binary mode	3.2.7		x				
Obey Telnet cmds in binary mode	3.2.7		x				
End-of-Line, CR NUL in binary mode	3.2.7						x
End-of-line							
EOL at server same as local EOL	3.3.1	2	x				
ASCII svr accept CR LF/CR NUL	3.3.1		x				
User Telnet send CR LF/CR NUL/LF	3.3.1	2	x				
ASCII user select CR LF/CR NUL	3.3.1	2		x			
User Telnet default mode is CR LF	3.3.1	2		x			
Non-interactive uses CR LF for EOL	3.3.1		x				
User Telnet interface							
Input & output all 7-bit characters	3.4.1			x			
Bypass local op sys interpretation	3.4.1			x			
Escape character	3.4.1	2	x				
User-settable escape character	3.4.1			x			
Escape to enter 8-bit values	3.4.1	1			x		
Can input IP, AO, AYT	3.4.2	3	x				
Can input EC, EL, Break	3.4.2	3		x			
Report TCP conn. errors to user	3.4.3	2		x			
Optional non-default contact port	3.4.4	3		x			
Can spec: output flush when IP sent	3.4.5			x			
Can manually restore output mode	3.4.5	1		x			

3.2 FILE TRANSFER

Three protocols are in general use for file transfer in IPS implementations:

- **FTP - File Transfer Protocol.**
  This protocol, using a reliable transport layer service (usually TCP), provides reliable file transfer including some level of independence from the differences between the two file systems involved in the transfer.
- **TFTP - Trivial File Transfer Protocol.**
  This is a basic file transfer protocol, using an datagram transport service (usually UDP), providing no authentication or other sophisticated user-level features. It has application in environments which cannot support the sophistication of the full FTP, in particular it is used for the remote booting of operating systems and network software.
- **rcp - Remote File Copy Utility.**
  This is a networked version of the UNIX *cp* file copy utility, part of the BSD implementation of TCP/IP. It has since been ported to most UNIX derivative systems and even some unrelated systems. It implements the characteristic command line syntax of the UNIX system and provides support for trusted hosts.

As discussed in **Section 1.2**, **Scope**, *rcp* and the other ''r'' commands are not covered here due to the lack of any formal specification of the protocols used.

The subsequent subsections cover the FTP and TFTP protocols, listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Requirements Summary table from [HR-APPS] is reproduced here with an additional column which highlights the areas of divergence.

3.2.1 FTP Requirements

For the FTP protocol, the requirements are specified by the RFC ''Host Requirements - Application and Support'' [HR-APPS] and by the base RFC RFC-959, ''File Transfer Protocol'' [FT-FTP].

In general, FTP implementations provide a basic STREAM mode, transfer of TYPE ASCII or TYPE Binary files. Block and Compressed modes and other file types are not supported. Most FTP commands are supported including those for ''third-party'' transfers between two remote hosts. The FTP utilities provide configurable access control per host and anonymous FTP plus conversion functions such as filename conversion and carriage return stripping. Most servers do not provide access to any non-standard features in their implementations and consequently have not provided the SITE command.

Pre-4.3 BSD implementations are slightly less complete, implementing fewer commands, specifically not third-party transfers and ''store-unique''.

More advanced versions of the BSD FTP source code are now available, giving an indication of the capabilities of the version to be included in the 4.4 BSD distribution. Among the possible enhancements are: an implementation of the RESTart command, based upon a restart marker which is a byte offset within the file being transferred; provision of access to some commands (*umask*, *idle*, *chmod* and *help*) via the SITE command; compile-time selection of Type-of-Service for the underlying transport service

(allowing selection of a high-bandwidth or low-delay path). It is likely that some up-to-date IPS implementations may already include some of these features.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the Requirements Summary table, the reference given directs the reader to the appropriate requirements RFC section. Where a requirement is supported other than in the ''pre-4.3 BSD'' implementations it is indicated as such in the Requirements Summary table and it is not included here.

- **4.1.2.2 - SHOULD implement TYPE T if same as TYPE N**
  TYPE T ''Telnet'' file type is not supported.
- **4.1.2.4 - File/Record transform SHOULD be invertible if poss.**
  Only the FILE file structure is generally supported.
- **4.1.2.9 - STOU cmd MUST return pathname as specified.**
  The Store Unique command is not implemented by pre-4.3 BSD products. Where it is, the file name is returned at the completion of the transfer using the 226 command and has the format: "226 Transfer complete (unique file name: *pppp*)".
- **4.1.2.11 - Server-FTP MUST send only correct reply format.**
  This requirement is generally observed, however there are a few ''bugs'' in current implementations which violate it and require the FTP user to take action to re-synchronise the connection.
- **4.1.2.11 - Server-FTP SHOULD use defined reply code if poss.**
  FTP servers appear to obey the rules for the meaning of the first digit of the reply. In general the specified codes are generated as replies, however the number of cases where a different code is sent are sufficiently numerous to make it unwise to rely on the subsequent digits of the reply code.
- **4.1.2.11 - User-FTP SHOULD NOT handle 421 reply specially.**
  User-FTPs based upon the 4.3 BSD code close the control connection in response to a 421 code from a Server-FTP in violation of this requirement.
- **4.1.2.12 - Server-FTP MUST handle Telnet options.**
  This requirement is observed, however implementations based upon the 4.3 BSD code may incorrectly reply to a DO option with a DONT option.
- **4.1.3.1 - SHOULD handle ''Experimental'' directory cmds.**
  There is some support for automatic fallback to the ''X'' version of a command when the standardised version is rejected. In at least one implementation, the ''X'' commands appear on the menu as alternatives to be tried if a regular command is rejected.
- **4.1.2.2 - SHOULD support TYPE ASCII - Telnet (AT) - if same as AN.**
  As already discussed, the AT TYPE is not supported.
- **RFC-959 3.1.1.5.2 - MAY support TYPE ASCII - carriage control.**
  This TYPE is not supported.
- **RFC-959 3.1.1.2 - MAY support TYPE EBCDIC (any form).**
  This TYPE is not supported.

- **4.1.2.1 - MAY support TYPE LOCAL m.**
  This TYPE is not supported.
- **RFC-958 3.4.2 - SHOULD NOT support MODE BLOCK.**
  Neither BLOCK nor COMPRESS MODEs are supported.
- **4.1.2.13 - MUST support STRUCTURE record.**
  There is only isolated support for STRUCTURE record.
- **4.1.2.13 and RFC-959 5.3.1 - Support commands:**
  The following commands are NOT supported:
  - SMNT - STRUCTURE MOUNT
  - REIN - REINITIALIZE
  - REST - RESTART, modern implementations may support the RESTart function in stream MODE, based upon an assumed block size of 1 byte. This means that no checkpoints are required, restart is based upon a byte-offset within the file.
  - STAT - STATUS (MUST)
  - SYST - SYSTEM (MUST)
  - SITE - SITE PARAMETERS, this command is not supported for the simple reason that current implementations do not support any non-standard features that would require it.

  In addition, there is wide but not complete support for the following commands:
  - STOU - STORE UNIQUE
  - PASV - PASSIVE DATA CONNECTION (MUST)
- **4.1.4.4 - SHOULD maintain synchronisation with server.**
  No automatic mechanism is provided. However, the ABORT command allows the FTP user to force the User-FTP and Server-FTP back into sync.

3.2.2 FTP Requirements Summary

This subsection reproduces the FTP Requirements Summary given in the RFC [HR-APPS] with an additional ‘‘XGIPS’’ column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.2.1**, **FTP Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 3.2.1**, **FTP Requirements**, for clarification.

3 Requirement not supported by ‘‘pre-4.3’’ implementations, see the introduction to **Section 3.2.1**, **FTP Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Implement TYPE T if same as TYPE N	4.1.2.2	1		x			
File/Rec. transform invertible	4.1.2.4	1		x			
Send PORT cmd for stream mode	4.1.2.5			x			
Server-FTP implement PASV	4.1.2.6	3	x				
PASV is per-transfer	4.1.2.6	3	x				
NLST reply usable in RETR cmds	4.1.2.7		x				
Implied type for LIST and NLST	4.1.2.7	3		x			
SITE cmd for non-standard features	4.1.2.8			x			
STOU cmd return path as spec.	4.1.2.9	1	x				
Use TCP READ boundaries on ctrl conn	4.1.2.10						x
S-FTP send only correct reply format	4.1.2.11	2	x				
S-FTP use defined reply code if poss	4.1.2.11	2		x			
New reply code follow Section 4.2	4.1.2.11				x		
U-FTP use only high digit of reply	4.1.2.11			x			
U-FTP handle multi-line replies	4.1.2.11		x				
U-FTP handle 421 reply specially	4.1.2.11	2					x
Def data port same addr as ctl conn	4.1.2.12		x				
U-FTP send Telnet cmds exc. SYNC/IP	4.1.2.12						x
User-FTP negotiate Telnet options	4.1.2.12						x
Server-FTP handle Telnet options	4.1.2.12	2	x				
Handle ‘‘experimental’’ directory cmds	4.1.3.1	2		x			
Idle timeout in server-FTP	4.1.3.2			x			
Configurable idle timeout	4.1.3.2			x			
Rcvr checkpoint data at Rest Marker	4.1.3.4			x			
Sender assume 110 replies are synch	4.1.3.4						x
Support TYPE:							
ASCII-Non-Print (AN)	4.1.2.13		x				
ASCII-T’net (AT) - if same as AN	4.1.2.2	1		x			
ASCII-Carriage Control (AC)	RFC 959	1			x		
EBCDIC - (any form)	RFC 959	1			x		
IMAGE	4.1.2.1		x				
LOCAL 8	4.1.2.1		x				
LOCAL m	4.1.2.1	1			x		
Support MODE:							
Stream	4.1.2.13		x				
Block	RFC 959	1			x		
Support STRUCTURE:							
File	4.1.4.13		x				
Record	4.1.4.13	2	x				
Page	4.1.4.3	-				x	
Support commands:							
USER	4.1.2.13		x				
PASS	4.1.2.13		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
ACCT	4.1.2.13		x				
CWD	4.1.2.13		x				
CDUP	4.1.2.13		x				
SMNT	RFC 959	1			x		
REIN	RFC 959	1			x		
QUIT	4.1.2.13		x				
PORT	4.1.2.13		x				
PASV	4.1.2.6	2	x				
TYPE	4.1.2.13		x				
STRU	4.1.2.13		x				
MODE	4.1.2.13		x				
RETR	4.1.2.13		x				
STOR	4.1.2.13		x				
STOU	RFC 959	2			x		
APPE	4.1.2.13		x				
ALLO	RFC 959				x		
REST	RFC 959	1			x		
RNFR	4.1.2.13		x				
RNTO	4.1.2.13		x				
ABOR	RFC 959				x		
DELE	4.1.2.13		x				
RMD	4.1.2.13		x				
MKD	4.1.2.13		x				
PWD	4.1.2.13		x				
LIST	4.1.2.13		x				
NLST	4.1.2.13		x				
SITE	4.1.2.8				x		
STAT	4.1.2.13	1	x				
SYST	4.1.2.13	1	x				
HELP	4.1.2.13		x				
NOOP	4.1.2.13		x				
User Interface:							
Arbitrary pathnames	4.1.4.1		x				
Implement ‘‘QUOTE’’ command	4.1.4.2	3	x				
Transfer control cmnds immediately	4.1.4.2			x			
Display error messages to user	4.1.4.3			x			
- Verbose mode	4.1.4.3			x			
Maintain synch with server	4.1.4.4	2		x			

3.2.3 TFTP Requirements

For the TFTP protocol, the requirements are specified by the RFC ‘‘Host Requirements - Application and Support’’ and by the base RFC RFC-783.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **4.2.3.1 - MUST Fix Sorcerer's Apprentice Syndrome.**
  Fix not implemented.
- **4.2.3.3 - Transfer Modes: MAY support extensions.**
  No extensions are provided.
- **4.2.3.2 - MUST use adaptive timeout.**
  Implementations use a fixed timeout interval.
- **4.2.3.4 - SHOULD provide configurable access control.**
  Some implementations allow the accessible portion of the directory tree to be limited. Generally, implementations give TFTP access to any file readable by ''Other'' users.
- **4.2.3.5 - SHOULD silently ignore broadcast request.**
  In general TFTP daemons DO respond to broadcast requests. This no-conformance is justified as a method by which diskless devices such as routers can locate a server to download its boot program or configuration file. Clearly servers must continue to respond to broadcast requests as long such devices are to be supported.

3.2.4 TFTP Requirements Summary

This subsection reproduces the TFTP Requirements Summary given in the RFC [HR-APPS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.2.3**, **TFTP Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 3.2.3**, **TFTP Requirements**, for clarification.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Fix Sorcerer's Apprentice Syndrome	4.2.3.1	1	x				
Transfer Modes							
- netascii	RFC-783		x				
- octet	RFC-783		x				
- mail	4.2.2.1					x	
- extensions	4.2.3.3	1			x		
Use adaptive timeout	4.2.3.2	1	x				
Configurable access control	4.2.3.4	2		x			
Silently ignore b'cast request	4.2.3.5	1		x			

3.3 ELECTRONIC MAIL

Within the IPS, electronic mail (email) is supported by two protocols:

- **SMTP - Simple Mail Transfer Protocol.**
  This protocol uses the information on a message's ''envelope'' to transfer it from the sender to the intended recipients. The protocol uses a ''reliable ordered data stream channel'' such as that provided by TCP to forward a message to the next SMTP server on the path to the eventual recipient. This protocol has no knowledge of the message's contents and includes a simple transparency mechanism to protect those contents from accidental interpretation.

- **RFC-822 - Standard for the Format of ARPA Internet Text Messages.**
  This standard defines how the contents of electronic mail messages are to be interpreted, defining the syntax for header fields such as those which specify the sender and recipient of the message.

The architecture for mail-handling systems is generally described using the terms borrowed from the specification of the CCITT X.400 standard for email. In particular the term ''Message Transfer Agent'' (MTA) is used to describe the component responsible for delivering a message from the sender to the recipient. The term ''User Agent'' is used to describe the component responsible for composing and reading mail messages.

For UNIX and derivative systems this architectural distinction is blurred by ''mailer programs'' which implement the MTA functions including the SMTP protocol along with some of the User Agent functions such as extraction of envelope information. There are many such programs available, some of them are proprietary, offering gateways into the supplier's own email system, there are also many public domain mailer programs. In the production of this guide, two mailer programs were considered:

- **Sendmail - the Berkeley 4.3 BSD mailer program.**
  This mailer is supported by most IPS implementations for UNIX and derivative systems and is thus used widely.

- **MMDF-II - Multi-channel Memo Distribution Facility.**
  This is public-domain software, produced by the academic community. MMDF is widely used and is thus chosen as representative of the many public domain mailers.

The remaining User Agent functions, presentation of messages and message composition, are implemented by a wide range of programs with diverse capabilities from the very basic *mail* utility to sophisticated screen-based mail management programs.

The subsequent subsections cover the two MAIL protocols, listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Requirements Summary Table from [HR-APPS] is reproduced here with an additional column which highlights the areas of divergence.

3.3.1 Electronic Mail Requirements

For electronic mail, the requirements are defined by the RFC ''Host Requirements - Application and Support'' [HR-APPS] and by the base RFCs RFC-821, ''Simple Mail Transfer protocol'' [EM-SMTP] and RFC-822, ''Standard for the Format of ARPA Internet Text Messages'' [EM-TEXT].

Both Sendmail and MMDF-II conform closely to the basic requirements for message formatting and transfer. The version of Sendmail distributed with 4.3 BSD is unsophisticated in its handling of unavailable hosts and uses the *gethostbyname* interface for host name resolution. The latest version of Sendmail in common use is Sendmail-5.61, the state-of-the-art version is Sendmail-5.64. These later versions provide better handling of dead hosts and make fuller use of the DNS including accessing ''MX'' records. The original version of MMDF-II did not interface to the DNS for resolving domain names, relying instead on a static host table. Later versions may use the DNS for domain name resolution.

The functionality of Sendmail is highly influenced by the configuration file which controls many aspects of the utility, particularly address processing. This section reflects the behaviour of Sendmail using the default configuration file, considerable changes may be effected by changing the rules contained there.

The RFCs 821 and 822 specifically prohibit 8-bit data in messages, the basic version of Sendmail consequently strips the high bit. The Host Requirements provide no guidance on this subject; however, many products have removed this restriction to allow international character sets to be transmitted in message headers and bodies. Unfortunately, no standard exists for defining the ''code-set'' to be used in interpreting the data in either part of the message although ''X-'' and the ''Content-Type:'' fields have been used for this purpose.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **5.2.3 - EXPN, VRFY support MAY be configurable.**
  No configuration switch is provided to inhibit these functions.
- **5.2.4 - MAY implement SEND, SOML, SAML.**
  These commands are not implemented.
- **5.2.5 - MAY verify HELO parameter.**
  Sendmail implementation makes no attempt at HELO verification. MMDF-II implements a strict mode where the supplied domain name is verified.
- **5.2.5 - MUST NOT refuse message with bad HELO.**
  When MMDF-II is configured to verify HELO messages it will reject lookup failures with a ''421 <*myname*>: Your name <*domain-name*> unknown to us.''
- **5.2.6 - SHOULD attempt to deliver source routed message directly.**
  No direct delivery is attempted using the default configuration file. It is possible to change the configuration file to re-write destination addresses for direct delivery. However, if this is done then Sendmail will not fall back to source routing if direct delivery fails.
- **5.2.8 - Received: line SHOULD include domain literal.**
  Domain literal not included, Received: line reports both the HELO name and the name derived from the source IP address of the connection.
- **5.2.10 - SHOULD send reply text from RFC-821 when appropriate.**
  In general, the ''standard'' reply texts are not sent.

- **5.3.2 - SHOULD wait at least 5 mins for next sender cmd.**
  Sendmail implements a configurable timeout on any read, default value of 5 minutes. MMDF-II does not support command timeouts.
- **5.3.3 - MUST send error message using null return path.**
  MMDF-II does this, Sendmail action is controlled by its configuration file, by default it sends ''MAIL FROM: <*MAILER-DAEMON@...*>''.
- **5.2.4 - MAY implement SEND, SOML, SAML.**
  Commands not implemented.
- **5.2.5 - MUST send principal host name in HELO.**
  host and domain name sent in HELO messages is configurable and is not validated.
- **5.2.6 - SHOULD NOT send explicit source route in RCPT TO:.**
  Sending of explicit source routes is fully supported, there is no provision to inhibit such routes.
- **5.2.10 - SHOULD use only high-digit of reply code when poss.**
  MMDF-II does a switch on full reply code to determine reply action.
- **5.3.1.1 - SHOULD retry once per each queued dest host.**
  MMDF-II implements check to avoid retries to dead hosts. The original Sendmail does not implement such checks, later versions may implement this performance enhancement.
- **5.3.1.1 - Sender MAY support multiple concurrent transactions.**
  MMDF-II allows multiple channels to be configured for sending mail. Sendmail processes mail queue sequentially.
- **5.3.1/5.3.2 - SHOULD implement per-command timeouts.**
  Sendmail implements a configurable (default 5 minutes) timeout on all reads. The timeout is across the board, the tailored timeouts are not provided. MMDF-II does not implement any activity timeouts.
- **5.3.2 - SHOULD implement recommended timeout values.**
  See the previous point ''Implement per-command timeouts''.
- **5.3.4 - MUST try alternate addrs in order.**
  MMDF-II allows multiple mappings to be specified in its configuration tables. Sendmail does not allow alternates.
- **5.3.4 - MAY implement configurable limit on alternate tries.**
  The MMDF-II limit on the number of alternative destinations is not configurable.
- **5.3.4 - SHOULD split load across equal MX alternates.**
  MMDF-II always tries alternates in the order specified, no rotation.
- **5.3.5 - MUST use the Domain Name System.**
  The 4.3 BSD Sendmail does use the DNS but only via the dumb *gethostbyname* routine, this does not give access to MX records. Later versions use a full interface to DNS. MMDF-II supports domain-style names via configuration table lookups. Later versions of MMDF MAY support DNS.
- **5.2.5 - MUST support MX records.**
  Later version of Sendmail which use a full interface to DNS support MX records.

- **5.2.6 - SHOULD NOT allow user to enter <route> address.**
  Generally, user agent programs do not restrict address syntax, however the MMDF-II ''msg'' and ''send'' restrict remote addresses to mbox@domain.
- **5.2.13 - MAY support RFC-1049 Content Type field.**
  This extension is not yet in wide use; however, some products use this field to signal alternate formats in the body part of the message.
- **5.2.14 - User Agent SHOULD use 4-digit year.**
  Mail utilities generally generate 2-digit years. Messages generated directly by Sendmail use 4-digit years.
- **5.2.14 - User Agent SHOULD generate numeric timezones.**
  Some mail utilities generate numeric timezones, however many still generate timezone names.

3.3.2 Electronic Mail Requirements Summary

This subsection reproduces the SMTP Requirements Summary given in the RFC [HR-APPS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.3.1**, **Electronic Mail Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 3.3.1**, **Electronic Mail Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

An entry in the ''FEATURE'' column marked with a double asterisk ''**'' represents an explicit requirement in the text of [HR-APPS] which has been missed from the appropriate summary table, it has been included here for completeness.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Implement VRFY	5.2.3		x				
Implement EXPN	5.2.3			x			
- EXPN, VRFY configurable	5.2.3	1			x		
Implement SEND, SOML, SAML	5.2.4	1			x		
Verify HELO parameter	5.2.5	2			x		
- Refuse message with bad HELO	5.2.5	2					x
Accept explicit src-route syntax in env.	5.2.6		x				
- Attempt direct delivery **	5.2.6	2		x			
Support ''postmaster''	5.2.7		x				
Process RCPT when received	5.2.7				x		

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
- Long delay of RCPT responses	5.2.7						x
Add Received: line	5.2.8		x				
- Received: line include domain literal	5.2.8	1		x			
Change previous Received: line	5.2.8						x
Pass Return-Path info (final deliv/gwy)	5.2.8		x				
Support empty reverse path	5.2.9		x				
Send only official reply codes	5.2.10			x			
Send text from RFC-821 when appropriate	5.2.10	1		x			
Delete ‘‘.’’ for transparency	5.2.11		x				
Accept and recognize self domain literal(s)	5.2.17		x				
Error message about error message	5.3.1						x
Keep pending listen on SMTP port	5.3.1.2			x			
Provide limit on recv concurrency	5.3.1.2				x		
Wait at least 5 mins for next sender cmd	5.3.2	2		x			
Avoidable delivery failure after ‘‘250 OK’’	5.3.3						x
Send error notification msg after accept	5.3.3		x				
- Send using null return path	5.3.3	2	x				
- Send to envelope return path	5.3.3			x			
- Send to null address	5.3.3						x
- Strip off explicit src route	5.3.3			x			
Minimize acceptance delay (RFC-1047)	5.3.3		x				
Sender-SMTP:							
Canonicalized domain names in MAIL, RCPT	5.2.2		x				
Implement SEND, SOML, SAML	5.2.4	1			x		
Send valid principal host name in HELO	5.2.5	1	x				
Send explicit source route in RCPT TO:	5.2.6	1				x	
Use only reply code to determine action	5.2.10		x				
Use only high digit of reply code when poss.	5.2.10	2		x			
Add ‘‘.’’ for transparency	5.2.11		x				
Retry messages after soft failure	5.3.1.1		x				
- Delay before retry	5.3.1.1		x				
- Configurable retry parameters	5.3.1.1		x				
- Retry once per each queued dest host	5.3.1.1	2	2	x			
Multiple RCPTs for same DATA	5.3.1.1			x			
Support multiple concurrent transactions	5.3.1.1	2			x		
- Provide limit on concurrency	5.3.1.1			x			
Timeouts on all activities	5.3.1		x				
- Per-command timeouts	5.3.2	1		x			
- Timeouts easily reconfigurable	5.3.2			x			
- Recommended times	5.3.2	1		x			
Try alternate addrs in order	5.3.4	2	x				
- Configurable limit on alternate tries	5.3.4	1			x		
- Try at least two alternates	5.3.4			x			
Load-split across equal MX alternates	5.3.4	1		x			
Use the Domain Name System	5.3.5	2	x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
- Support MX records	5.3.5	2	x				
- Use WKS records in MX processing	5.2.12	1				x	
Mail Forwarding:							
Alter existing header field(s)	5.2.6					x	
Implement relay function: 821/section 3.6	5.2.6				x		
- If not, deliver to RHS domain	5.2.6	-		x			
Interpret ‘‘local-part’’ of addr	5.2.16						x
Mailing Lists and Aliases:							
Support both	5.3.6			x			
Support mail list error to local admin.	5.3.6		x				
Mail Gateways:							
Embed foreign mail route in local-part	5.2.16				x		
Rewrite header fields when necessary	5.3.7				x		
Prepend Received: line	5.3.7		x				
Change existing Received: line	5.3.7						x
Accept full RFC-822 on Internet side	5.3.7			x			
Act on RFC-822 explicit source route	5.3.7				x		
Send only valid RFC-822 on Internet side	5.3.7		x				
Deliver error msgs to envelope addr	5.3.7			x			
Set env return path from err return addr	5.3.7			x			
User Agent - RFC-822							
Allow user to enter <route> address	5.2.6	2				x	
Support RFC-1049 Content Type field	5.2.13	2			x		
Use 4-digit years	5.2.14	2		x			
Generate numeric timezones	5.2.14	2		x			
Accept all timezones	5.2.14		x				
Use non-num timezones from RFC-822	5.2.14		x				
Omit phrase before route-addr	5.2.15				x		
Accept and parse dot.dec. domain literals	5.2.17		x				
Accept all RFC-822 address formats	5.2.18		x				
Generate invalid RFC-822 address format	5.2.18						x
Fully-qualified domain names in header	5.2.18		x				
Create explicit src route in header	5.2.19					x	
Accept explicit src route in header	5.2.19		x				
Send and recv at least 64KB messages	5.3.8		x				

3.4 SUPPORT SERVICES

As discussed in **Section 1.2**, **Scope**, out of the protocols addressed by the ''Support Services'' section of [HR-APPS] only the Domain Name System (DNS) name resolution protocol is covered by this guide.

The DNS primarily supplies a name and address lookup service to the IPS protocol suite. Other information concerning a host's configuration and MX records for mail delivery is also handled. The DNS implements a tree-structured name space including the concept of ''Zones of Authority'' to allow the responsibility for name space administration to be distributed. The protocol divides into two halves, ''resolvers'' which operate on behalf of clients to perform name and address lookups, and ''name servers'' which accept lookup requests and cooperate with other name servers to resolve them.

The next subsection covers the DNS protocol, listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. The Requirements Summary Table from [HR-APPS] is reproduced here with an additional column which highlights the areas of divergence.

3.4.1 DNS Requirements

The requirements for the Domain Name Service are defined by the RFC

- ''Host Requirements - Application and Support'' [HR-APPS]

and by the base RFCs

- RFC-1034, ''Domain Names - Concepts and Facilities'' [SS-DOMC], and
- RFC-1035, ''Domain Names - Implementation and Specification'' [SS-DOMI].

For UNIX and derivative systems, one DNS implementation is pre-eminent, the BIND program supplied with the Berkeley 4.3 BSD networking code (also known as *named* after its name server daemon). The BIND distributed with 4.3 BSD was version 4.3. It provides a simple resolver (known as a ''stub resolver''), implemented as a sub-routine library, which does not cache information and relies upon name servers to resolve any query fully. Only basic abbreviation facilities (local domain completion) are provided. The BIND *named* daemon provides a full functionality name server with a cache for information received from other name servers. most record types including MX records are supported, There is no support for TXT and NULL records.

Many products include more up-to-date versions and it is not unusual for a particular installation to run a version obtained from public domain software servers. The latest version in common use is BIND 4.8 which adds a number of enhancements including a more sophisticated resolver which supports broadcast requests for name server discovery and several heuristics for expanding abbreviated domain names.

Future versions of BIND may support caching of negative responses to avoid repeated requests for a non-existent name (such as when a widely accessed host changes its name or an obsolete ''network domain'' is phased-out). The experimental TEXT resource record is also likely to be supported by future versions.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the summary table, and the reference given directs the reader to the appropriate requirements RFC section.

- **6.1.2.1 - MUST properly handle RR with zero TTL**
  A Resource Record with a zero value TTL is cached but for a fixed, short period (5 minutes).
- **6.1.3.5 - MUST support all well-known, class-indep. types.**
  Neither TXT or NULL RR types are supported by BIND 4.3. The NULL Resource Record is supported in BIND 4.8. TXT Resource Records are currently only experimental, the latest versions of BIND are likely to support them.
- **6.1.3.1 - MAY implement a full-service resolver.**
  BIND implements a ‘‘stub’’ resolver.
- **6.1.3.1 - Stub resolver MAY implement local cache.**
  No local cache implemented.
- **6.1.3.4 - SHOULD sort multiple addresses by preference list.**
  Network preference lists are not implemented by BIND 4.3. Later versions of BIND do provide a resolver which sorts responses against a list of preferred network numbers. In addition, the BIND 4.8 name server sorts its responses to prefer local networks and networks from a ‘‘sort-list’’.
- **6.1.3.2 - SHOULD support TCP queries.**
  TCP queries are supported subject to the qualifications listed below.
- **6.1.3.2 - SHOULD try TCP if UDP answers are truncated.**
  The BIND 4.3 resolver supports the use of TCP for queries as a result of receiving a truncated response to a UDP query. Unless configured to ignore truncation entirely, the resolver will always retry using a TCP connection even when the truncation affects only the ‘‘Additional’’ Resource Records (i.e. the ‘‘Answer’’ records are complete). The BIND 4.3 name server supports incoming TCP requests but only uses TCP for making zone transfer requests.
- **6.1.3.2 - MAY limit TCP query resources.**
  Named does not monitor use of TCP queries.
- **6.1.3.2 - MUST NOT use truncated data as if it were not.**
  The BIND 4.3 resolver either re-requests truncated data using TCP to get the full response or passes back the full message with truncated indicator to the caller. The BIND 4.3 name server takes no notice of the truncated indicator in a response. It stores and passes-on the information as though it is complete.
- **6.1.3.2 - MAY make private agreement to use only TCP.**
  The resolver provides a flag which determines whether a particular query is to be made using TCP only. The BIND 4.3 name server does not support the use of TCP for queries to other name servers. Later versions of the name server code do support this feature.

- **6.1.3.2 - MAY support broadcast or multicast queries.**
  Use of broadcast/multicast for name server ''discovery'' is not supported by BIND 4.3. In later versions the name server does respond to broadcast requests, however, the resolver only uses unicast datagrams and must know the address of at least one name server.
- **6.1.3.2 - Server MUST ignore RD bit in b'cast/m'cast requests.**
  Named does not check the destination address of a request.
- **6.1.3.3 - SHOULD cache temporary failures.**
  Named does not cache such failures.
- **6.1.3.3 - SHOULD cache negative responses.**
  Named does not cache negative responses. Future versions of BIND are likely to support this feature.
- **6.1.3.3 - Retries SHOULD use exponential back-off.**
  The BIND 4.3 name server uses a fixed retry period and count. Later versions double the timeout on each subsequent retry.
- **6.1.3.3 - Client SHOULD handle Source Quench.**
  Source Quench messages are not visible to the resolver, due to the socket interface and UDP implementation.
- **6.1.3.3 - Server MAY ignore Source Quench.**
  See previous point.
- **6.1.4.3 - MAY provide name abbreviation facilities.**
  Later versions provide abbreviation lists in the resolver which match the entered name against a list of name/value pairs.
- **6.1.4.3 - MAY implement abbreviation search lists.**
  The BIND 4.3 resolver provides a simple form of abbreviation by appending the local domain name to any query which does not have a trailing dot.

3.4.2 DNS Requirements Summary

This subsection reproduces the Domain Name System Requirements Summary given in the RFC [HR-COMMS] with an additional ''XGIPS'' column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.4.1**, **DNS Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 3.4.1**, **DNS Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
General Issues:							
Implement DNS name-to-address conversion	6.1.1		x				
Implement DNS address-to-name conversion	6.1.1		x				
Support conversion using host table	6.1.1				x		
Properly handle RR with zero TTL	6.1.2.1	1	x				
Use QCLASS=* unnecessarily	6.1.2.2					x	
- Use QCLASS=IN for internet class	6.1.2.2		x				
Unused fields zero	6.1.2.3		x				
Use compression in responses	6.1.2.4		x				
Include config info in responses	6.1.2.5						x
Support all well-known, class-indep. types	6.1.3.5	2	x				
Easily expand type list	6.1.3.5			x			
Load all RR types (except MD and MF)	6.1.3.6		x				
Load MD or MF type	6.1.3.6						x
Operate when root servers, etc. unavailable	6.1.3.7		x				
Resolver Issues:							
Resolver support multiple concurrent requests	6.1.3.1			x			
Full-service resolver:	6.1.3.1	1			x		
- Local cache	6.1.3.1	-	x				
- Information in local cache times out	6.1.3.1	-	x				
- Configurable with starting info	6.1.3.1	-		x			
Stub resolver:	6.1.3.1				x		
- Use redundant recursive name servers	6.1.3.1		x				
- Local cache	6.1.3.1	1			x		
- Information in local cache times out	6.1.3.1	-	x				
Support for multi-homed hosts:							
- Sort multiple addresses by preference list	6.1.3.4	2		x			
Transport Protocols:							
Support UDP queries	6.1.3.2		x				
Support TCP queries	6.1.3.2	2		x			
- Send query using UDP first	6.1.3.2		x				
- Try TCP if UDP answers are truncated	6.1.3.2	2		x			
Name server limit TCP query resources	6.1.3.2	1			x		
- Punish unnecessary TCP query	6.1.3.2					x	
Use truncated data as if it were not	6.1.3.2	2					x
Private agreement to use TCP only	6.1.3.2	2			x		
Use TCP for zone transfers	6.1.3.2		x				
TCP usage not block UDP queries	6.1.3.2		x				
Support broadcast or multicast queries	6.1.3.2	1			x		
- RD bit set in query	6.1.3.2	-					x
- RD bit ignored by server if b'cast / m'cast	6.1.3.2	1	x				
- Send only as occasional probe for addr's	6.1.3.2	-		x			
Resource Usage:							
Transmission controls, per [DNS:2]	6.1.3.3		x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
- Finite bounds per request	6.1.3.3		x				
Failure after retries => soft error	6.1.3.3		x				
Cache temporary failures	6.1.3.3	1		x			
Cache negative responses	6.1.3.3	1		x			
Retries use exponential back-off	6.1.3.3	2		x			
- Upper, lower bounds	6.1.3.3	-		x			
Client handle Source Quench	6.1.3.3	1		x			
Server ignore Source Quench	6.1.3.3	1			x		
User Interface:							
All programs have access to DNS interface	6.1.4.2		x				
Able to request all info for given name	6.1.4.2		x				
Returns complete info or error	6.1.4.2		x				
Special interfaces	6.1.4.2				x		
- Name <-> Address translation	6.1.4.2		x				
Abbreviation facilities	6.1.4.3	2			x		
Convention for complete names	6.1.4.3		x				
Conversion exactly once	6.1.4.3		x				
Conversion in proper context	6.1.4.3		x				
Search list:	6.1.4.3	2			x		
- Administrator can disable	6.1.4.3			x			
- Prevention of excessive root queries	6.1.4.3		x				
- Both methods	6.1.4.3			x			

3.5 APPLICATION LAYER - GENERAL

The requirements RFC [HR-APPS] includes a section dealing with requirements applying to all applications. They mainly deal with issues concerning the Domain Name Service (DNS), IP addresses and the use of Type-of-Service (TOS).

3.5.1 General Application Requirements

The remainder of this section consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **2.1 - MUST allow host name to begin with a digit.**
  Some implementations, based on early versions of the Berkeley software may not allow hostnames with a leading digit.
- **2.1 - SHOULD allow host names of up to 255 characters.**
  Most implementations allow host names of up to 255 characters to be looked up in the hosts database or by DNS. However some may limit the name returned by the *hostname*( ) routine to 63 characters.
- **2.1 - SHOULD check syntactically for dotted-dec first.**
  In general utilities such as Telnet and FTP do this correctly, however at least one (TFTP) calls *gethostbyname* before *inet_addr*. Later versions of TFTP may have fixed this shortcoming.
- **2.2 - MUST cope with soft DNS errors.**
  Some of the interactive utilities such as Telnet access the DNS via the *gethostbyname* sub-routine call and do not check for ‘‘soft’’ lookup failures, thus they always report ‘‘host unknown’’. Where it is important to detect soft errors (e.g. Mail applications) such failures are likely to be processed correctly.
- **2.2 - UDP reply address SHOULD be specific destination of request.**
  The reply source address is generally the ‘‘primary’’ address of the server host. An application is not informed of the specific-destination address of a datagram.
- **2.4 - Application MUST specify appropriate TOS value.**
  The socket interface / TCP implementation does not allow an application to specify TOS values.

3.5.2 General Application Requirements Summary

This subsection reproduces the General Application Requirements Summary given in the RFC [HR-COMMS] with an additional ‘‘XGIPS’’ column. A note in this column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 3.5.1**, **General Application Requirements** for possible clarification.

2 Requirement supported with some qualification, see **Section 3.5.1**, **General Application Requirements** for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
User Interfaces:							
Allow host name to begin with digit	2.1		x				
Host names of up to 63 characters	2.1		x				
Host names of up to 255 characters	2.1	2		x			
Support dotted-decimal host numbers	2.1			x			
Check syntactically for dotted-dec first	2.1	2		x			
Map domain names per Section 6.1	2.2		x				
Cope with soft DNS errors	2.2	2	x				
- Reasonable interval between retries	2.2		x				
- Allow for long outages	2.2		x				
Expect WKS records to be available	2.2					x	
Try multiple addrs for multihomed host	2.3			x			
UDP reply src addr is spec. dest of req	2.3	1		x			
Use same IP addr for related TCP conns	2.3			x			
Specify appropriate TOS values	2.4	1	x				
- TOS values configurable	2.4	-	x				
- Unused TOS bits zero	2.4	-	x				

Chapter 4

Gateway Systems

This chapter addresses the requirements for IPS implementations when acting in the role of a gateway between a number of networks. As discussed in **Section 1.2**, **Scope**, UNIX and derivative systems generally act as a gateway between LANs or perhaps between a cluster of LANs and a wide-area network. In particular this excludes systems acting as gateways between WANs.

4.1 GATEWAY SYSTEMS

The Gateway Requirements RFC [GS-GR] addresses specific IP options and features required for packet forwarding, routing protocols and network interfaces. The requirements contain an extensive gateways algorithms section giving guidance for designers of new routing protocols, these requirements are not considered here. The requirements also cover gateway operation and maintenance in some detail. These requirements are not considered here as they are more appropriate to a discussion of management protocols.

The subsequent sections cover the gateway requirements specified by [GS-GR] listing the requirements and protocol features not generally supported by IPS implementations for UNIX and derivative systems. A requirements summary table like those provided for the Host Requirements is provided to assist the reader.

4.1.1 Gateway Requirements

The requirements for a system acting as an internet gateway are defined by RFC-1009 ‘‘Requirements for Internet Gateways’’ [GS-GR] which in turn references the base RFC for the IP protocol RFC-791, ‘‘Internet Protocol’’ [IL-IP] and many RFCs for the routing and link-layer protocols.

For routing protocols, there is general support for RIP due to its inclusion in the BSD 4.3 network implementation. There is some support for the ‘‘gated’’ routing daemon which provides an implementation of RIP, HELLO and EGP in a single directly coupled package.

For network interfaces there is universal support for an Ethernet interface but not for IEEE 802.2 encapsulation. There is isolated support for PDN X.25 access. Most implementations support a serial line interface, based upon the SLIP protocol, however this protocol was designed as a ‘‘low-tech’’ method for personal computer connection and is not appropriate for connection of gateways over high-speed links, in the future it is likely to be replaced by PPP as the link layer protocol for running IP over serial lines due to superior performance.

The remainder of this subsection consists of a list of requirements which current implementations generally do not meet. Each bullet point corresponds to an entry in the summary table, the reference given directs the reader to the appropriate requirements RFC section.

- **2.1 - MUST ignore datagrams with class D/E addresses.**
  These addresses will be mistakenly treated as class ‘‘C’’ addresses and may be forwarded or dropped with ICMP error reply.

- **2.2 - MUST NOT send ICMP error for ICMP error, b'cast or m'cast d'grams.**
  This requirement is honoured other than as noted in the section ''Internet Layer'' (Sect 4.3.1, page xxx - bullets 3.2.2).
- **2.2.1 - MUST send HOST UNREACH unless dest. network unreachable.**
  The gateway will always send NET UNREACH.
- **2.2.3 - MUST send QUENCH when congested.**
  No mechanism is provided for sending quench messages when datagrams cannot be accepted due to a lack of buffers. Quench is sent when forwarded datagram cannot be fragmented due to lack of buffers.
- **2.2.7 - SHOULD use TSP or NTP to establish local clock.**
  There is support for both protocols although neither is commonly included as part of IPS implementations. Public domain versions of NTP are available and it is in general used within the Internet for time synchronisation.
- **2.2.7 - If no time-base SHOULD flag TIMESTAMP as non-std.**
  Even where there is no sophisticated synchronisation protocol the TIMESTAMP message flags the time supplied as ''standard''.
- **2.3 - MAY support the EGP protocol.**
  EGP is supported by implementations which provide the ''gated'' routing daemon.
- **2.3 - Distance metric and polling interval SHOULD be configurable.**
  Polling interval is fixed for ''gated''.
- **2.6.2 - MAY support SPF routing protocols.**
  A standard for an ''Shortest-Path-First'' routing protocol is relatively recent and is not yet supported.
- **2.6.3 - MAY support the RIP protocol.**
  This protocol is widely supported via the ''routed'' daemon distributed with the BSD 4.3 networking software. The ''gated'' daemon also supports it. Both implement ''a variant'' of the RIP protocol, unfortunately there is no indication of what this variant is.
- **2.6.3 - RIP ''hold-down'' MUST be configurable.**
  Neither ''routed'' not ''gated'' provide a configurable value for this.
- **2.6.4 - MAY support the HELLO protocol.**
  HELLO is supported by implementations which provide the ''gated'' routing daemon.
- **2.8 - MUST drop type ''D'' packets silently.**
  See comment above regarding address formats.
- **3.1 - MAY support PDN access via X.25.**
  There is only limited support for X.25 connection to an X.25-based public data network. Where it is supported it is often supplied as a separately purchased product.
- **3.2 - MAY support ARPANET access via the 1822 protocol.**
  There is no support for this type of interface.
- **3.3 - MAY support DDN access via X.25.**
  There is only limited support for X.25 connection to the DDN network. Where it is

supported it is often supplied as a separately purchased product. There are few connections actually installed.

- **3.4 - MAY support access to Ethernet/IEEE 802 networks.**
  All systems support the standard access using ‘‘Ethernet encapsulation’’ ie placing the EtherType field in the Ethernet packet header. As yet there is little support transmission of IP packets using 802.2 encapsulation.
- **3.5 - MAY support Serial Line Framing Protocol (SLIP).**
  Most implementations appear to support the SLIP protocol although documentation and configuration details are scarce.
- **3.5 - SHOULD support data compression on serial lines.**
  There is no support for any data compression algorithm.
- **3.5 - MAY implement ‘‘half-gateway’’ protocol for serial lines.**
  There is no support for ‘‘half-gateways’’. Some dedicated router products use the half-gateway technique for inter-LAN links.

4.1.2 Gateway Requirements Summary

This subsection reproduces a summary table in the style of those included in the Host Requirements RFCs, generated from the requirements listed in the Gateway Requirements RFC [GS-GR]. A note in the additional ‘‘XGIPS’’ column indicates some level of non-compliance and should be interpreted as follows:

1 Requirement generally not supported, see **Section 4.1.1**, **Gateway Requirements**, for possible clarification.

2 Requirement supported with some qualification, see **Section 4.1.1**, **Gateway Requirements**, for clarification.

\- Requirement not applicable, see a preceding list item.

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
IP Requirements:							
Implement IP and Subnets	2.1		x				
Ignore TOS, Security and Stream ID opts	2.1				x		
Support SSRR, LSRR, RR & TS options	2.1		x				
Implement fragmentation	2.1		x				
Implement fragment reassembly	2.1		x				
- Reassembly buffer > MAX(MTU)	2.1		x				
Ignore i/f addr for gateway datagrams	2.1		x				
Ignore d'grams with class D/E addresses	2.1	1	x				
ICMP Requirements:							
ICMP errors for ICMP error or b'cast	2.2	2					x
Send HOST UNREACH unless dest net unreach	2.2.1	1	x				
Send only HOST(/TOS) REDIRECTS	2.2.2			x			
QUENCH when d'gram dropped due to congestion	2.2.3	2	x				
- Implement alg to limit QUENCH messages	2.2.3	-	x				
- limit parameters configurable	2.2.3	-	x				
- Allow QUENCH to be disabled	2.2.3	-		x			
Send TIME EXCEEDED when TTL -> 0	2.2.4				x		
Send TIME EXCEEDED on reassembly timeout	2.2.4				x		
Send PARAM PROB on other errors	2.2.5				x		
Support ADDRESS MASK a-la RFC-950	2.2.6		x				
Support TIME STAMP messages	2.2.7		x				
- Support TSP or NTP clock synchronization	2.2.7	2		x			
- If no time-base then flag as non-std	2.2.7	1		x			
Support INFO messages	2.2.8				x		
Support ECHO messages	2.2.9		x				
- Use Src Route for reply	2.2.9			x			
EGP Requirements:							
Support EGP	2.3	2			x		
Couple EGP to interior routing protocols	2.3		x				
Compare distances outside a single AS	2.3						x
configurable distance and polling param	2.3	2		x			
Interior Gateway Protocol Requirements:							
Support GGP	2.6.1					x	
Support SPF type protocol	2.6.2	1			x		
Support RIP protocol	2.6.3	2			x		
- Avoid RIP ''counting to infinity'' bug	2.6.3		x				
- configurable hold-down period	2.6.3	1	x				
Support HELLO protocol	2.6.4	2			x		
Support IGMP	2.8					x	
- Drop type ''D'' packets silently	2.8	1	x				

FEATURE	REQ RFC SECTION	XGIPS	MUST	SHLD	MAY	SHLD NOT	MUST NOT
Network Interface Requirement:							
Send/Receive d'grams <= MAX(MTU)	3		x				
Access PDN via X.25	3.1	2			x		
- conform to RFC-877	3.1		x				
Access ARPANET via 1822	3.2	1			x		
Access ARPANET via DDN std X.25	3.3	1			x		
- report dest host failure using UNREACH	3.3	-	x				
Access Ethernet/IEEE 802	3.4	2			x		
- Implement RFC-894 for Ethernet	3.4		x				
- Implement RFC-1042 for IEEE 802	3.4	-	x				
Serial Line Protocols:							
- X.25 LAPB	3.5	1			x		
- HDLC Framing	3.5	1			x		
- Xerox Sync. Point-to-Point	3.5	1			x		
- Serial Line Framing Protocol	3.5	2			x		
- Employ data compression algorithm	3.5	1		x			
- Conserve network numbers	3.5			x			
- Implement ''half gateway'' protocol	3.5	1			x		
Gateway Algorithms:							
Not covered by this guide.							
Operation and Maintenance Requirements:							
Not covered by this guide.							

Gateway Systems

Appendix A

Glossary

This glossary covers the conventions and terminology used in this guide. References to other documents are marked by a key within square braces (e.g. [HR-APPS]). The section **Referenced Documents** gives the meaning of the reference keys.

The following list defines the terms and abbreviations used in this guide. It does not attempt to cover many of those used in the requirements lists and tables, here the user is referred to the associated RFCs and other specifications. Some of the definitions given here repeat or quote from those given in the host requirements RFCs.

- **API**
  Application Programming Interface.
- **ARP**
  Address Resolution Protocol. A protocol for mapping an internet address into an address understood by a link-layer protocol (usually Ethernet).
- **ARPANET**
  A network set up by *DARPA* in the early 1970s for research into packet-switched networks.
- **Autonomous System**
  An autonomous system is a set of gateways under the control of a single operations and maintenance organisation and employing common routing protocols among themselves.
- **BIND**
  Berkeley Internet Name Domain. An implementation of *DNS*, part of the 4BSD UNIX-derivative system.
- **BOOTP**
  Boot Protocol. A protocol used to communicate configuration data such as internet addresses. May be used by diskless workstations at boot time.
- **BSD**
  Berkeley Software Distribution. Distributed by the University of California at Berkeley. A UNIX-derivative operating system, used as the starting point for a number of commercial UNIX-derived systems. It includes an implementation of IPS which has formed the basis of many commercial IPS products.
- **Daemon**
  In UNIX terminology a daemon is a server process, thus the Telnet daemon is a server which listens for and supports Telnet connections.
- **DARPA**
  Defence Advanced Research Projects Agency, formerly known as ARPA. Major sponsor of the development of the IPS.

- **Datagram**
  The host requirements RFCs defines the term IP Datagram as ‘‘the unit of end-to-end transmission in the IP protocol’’. In this guide, as in the RFCs, the term datagram is equivalent to IP Datagram.
- **DNS**
  Domain Name Service. A name and address lookup protocol.
- **EGP**
  Exterior Gateway Protocol. A protocol for propagating routing information outside of an *Autonomous System*.
- **Email**
  Electronic Mail
- **FEP**
  See *Front-End-Processor*.
- **FTP**
  File Transfer Protocol. Uses the services of a connection-oriented transport protocol (usually TCP).
- **Front-End-Processor (FEP)**
  An intelligent peripheral device which supports some or all of a communication protocol, removing processing load from its host system and partitioning communications from other host services.
- **Gateway**
  Within this guide and the host requirements RFCs the term Gateway or Gateway System means an IP-level router. This is in contrast to a *Host* system.
- **HELLO**
  One of a number of routing protocols used within an *Autonomous System*.
- **Host**
  Within this guide and the Host Requirements RFCs the term Host refers to a system supporting the transport-layer and application-layer protocols (i.e. supporting users or server functions). This is in contrast to a *Gateway*.
- **IAB**
  See *Internet Activities Board*.
- **ICMP**
  Internet Control Message Protocol. Logically part of the *IP* protocol, this protocol provides control functions such as error reporting and first-hop gateway redirection.
- **Internet**
  The collective name for the internetwork of research organisation networks of which *ARPANET* was a founding member. The Internet now comprising local, regional and backbone networks in the U.S., North America, and a number of other continents.
- **Internet Activities Board (IAB)**
  The coordinating committee for Internet design, engineering and management. The body that sets Internet standards (*RFCs*) and thus standards for the IPS.

- **Internet Protocol Suite (IPS)**
  The layered set of protocols used in an internet network. Also referred to as ‘‘TCP / IP’’.

- **IP**
  Internet Protocol. The connectionless internetwork service of the IPS. It provides no end-to-end delivery guarantees. It roughly corresponds to the network layer in the *OSI seven-layer model*.

- **IPS**
  See *Internet Protocol Suite*.

- **LLC**
  See *Logical Link Control*.

- **Logical Link Control (LLC)**
  The protocol used at the link layer for use on IEEE 802.*x* local-area networks, defined by IEEE 802.2 [LK-802.2]. It is used loosely to refer to LLC Type 1 which is the connectionless mode of operation for the LLC protocol.

- **Message Transfer Agent (MTA)**
  A component of the X.400 electronic mail standard. Responsible for transmitting and receiving messages.

- **MMDF**
  Multi-channel Memo Distributions Facility. A public domain electronic mail system.

- **MTA**
  See *Message Transfer Agent*.

- **NFS**
  Network File System. A suite of protocols for sharing files on a network by ‘‘mounting’’ portions of a remote system’s file space within the local directory hierarchy.

- **OSI seven-layer model**
  The ISO Reference Model for OSI (ISO7498). A conceptual model which provides a common basis for describing communications protocols and services.

- **Packet**
  The host requirements RFCs define the term IP packet as ‘‘the unit of data passed across the interface between the internet layer and the link layer’’. In this guide, as in the RFCs, the term packet is equivalent to IP packet.

- **PDN**
  See *Public Data Network*.

- **PPP**
  Point to Point Protocol. The likely successor to *SLIP* as the standard link layer protocol for running IP over serial lines.

- **Profile**
  The ISO standard TR-10000 defines a profile as ‘‘a set of one or more base standards and, where applicable, the identification of chosen classes, subsets, options and parameters of those base standards necessary for accomplishing a particular function’’.
  This guide, together with the requirements RFCs, is a profile of the IPS.

- **Public Data Network (PDN)**
  A data network (in this guide an X.25 packet switched network) provided by a national telecommunications authority.

- **R Protocols**
  rsh, *rlogin*, *rcp*. A set of protocols developed for the *BSD* UNIX-derivative, designed to provide UNIX-like services in a networked environment.
  The scope of this guide excludes these protocols.

- **RARP**
  Reverse ARP Protocol. A protocol for determining a system’s internet address from its link layer address. May be used by diskless workstations to obtain basic configuration information during booting.

- **RFC**
  Request For Comments. A document, issued by the *IAB*, used for specifying the protocols of the *IPS*.

- **RIP**
  Routing Information Protocol. One of a number of routing protocols used within an *Autonomous System*.

- **Rlogin**
  Remote login protocol. One of the *‘‘r’’ protocols*.

- **Sendmail**
  The 4.3 BSD mailer program.

- **SLIP**
  A simple link-layer protocol for serial lines.

- **SMTP**
  Simple Mail Transfer Protocol. The IPS protocol used for forwarding email messages.

- **SNAP**
  See *Sub-Network Access Protocol*.

- **Socket**
  For *TCP/UDP* a socket is a communications end point, defined by an internet address and a TCP/UDP port number. Within the *BSD* UNIX-derivative it is also the generic name of the protocol-independent inter-process communication interface.

- **STREAMS**
  The UNIX System V communications paradigm. It is a software architecture which allows a protocol stack to be implemented in a layered fashion.

- **Sub-Network Access Protocol (SNAP)**
  An extension to the *LLC* protocol, used to extend LLC addressing. IP uses SNAP to encapsulate a number of discrete protocols (currently IP and ARP) without requiring multiple Link Service Access Points.
- **SVID**
  System V Interface Definition. AT&T's definition of the interface to the services provided by UNIX System V.
- **TCP**
  Transport Control Protocol. The reliable, sequenced, byte stream transport protocol of the Internet Protocol Suite. Roughly corresponds to the Connection Oriented Transport Service at the transport layer in the *OSI seven-layer model*.
- **TCP/IP**
  A common name for the Internet Protocol Suite.
- **TFTP**
  Trivial File Transfer Protocol. Low-function file transfer, using a connection-less transport service (usually *UDP*). May be used by diskless workstations for down-loading system images at boot time.
- **TLI**
  See *Transport Layer Interface*.
- **TOS**
  See *Type of Service*.
- **Telnet**
  The remote login or virtual terminal protocol of the Internet Protocol Suite.
- **Transport Layer Interface (TLI)**
  The **SVID** protocol-independent interface to transport layer services.
- **Type of Service (TOS)**
  Used by an application or transport layer protocol to specify the handling requirements (bandwidth, priority, etc) for a message or session.
- **UA**
  See *User Agent*.
- **UDP**
  User Datagram Protocol. The connectionless transport layer protocol of the Internet Protocol Suite. Roughly corresponds to the Connectionless Transport Service at the Transport Layer in the *OSI seven-layer model*.
- **User Agent (UA)**
  A component of the X.400 electronic mail protocol. Responsible for accepting and delivering messages to the user.
- **X.25**
  A wide area network protocol defined by the CCITT and adopted by ISO. It operates at the network and link layers of the *OSI seven-layer model* and is used on many *PDNs*.
- **X.400**
  The electronic mail protocol, defined by CCITT and adopted by ISO.

- **XGIPS**
  X/Open Guide to the Internet Protocol Suite.
- **XTI**
  X/Open Transport Interface. X/Open's definition of an interface to the transport layer services, part of the X/Open Portability Guide (XPG). Based upon *TLI*.

Appendix B

RFC-1122, Host Requirements - Communications Layers

This appendix reproduces, in-full and un-edited, an RFC published by the Internet Engineering Task Force of the Internet Activities Board.

Network Working Group
Request for Comments: 1122

Internet Engineering Task Force
R. Braden, Editor

October 1989

Requirements for Internet Hosts -- Communication Layers

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This is one RFC of a pair that defines and discusses the requirements for Internet host software. This RFC covers the communications protocol layers: link layer, IP layer, and transport layer; its companion RFC-1123 covers the application and support protocols.

Table of Contents

1. INTRODUCTION

This document is one of a pair that defines and discusses the requirements for host system implementations of the Internet protocol suite. This RFC covers the communication protocol layers: link layer, IP layer, and transport layer. Its companion RFC, "Requirements for Internet Hosts -- Application and Support" [INTRO:1], covers the application layer protocols. This document should also be read in conjunction with "Requirements for Internet Gateways" [INTRO:2].

These documents are intended to provide guidance for vendors, implementors, and users of Internet communication software. They represent the consensus of a large body of technical experience and wisdom, contributed by the members of the Internet research and vendor communities.

This RFC enumerates standard protocols that a host connected to the Internet must use, and it incorporates by reference the RFCs and other documents describing the current specifications for these protocols. It corrects errors in the referenced documents and adds additional discussion and guidance for an implementor.

For each protocol, this document also contains an explicit set of requirements, recommendations, and options. The reader must understand that the list of requirements in this document is incomplete by itself; the complete set of requirements for an Internet host is primarily defined in the standard protocol specification documents, with the corrections, amendments, and supplements contained in this RFC.

A good-faith implementation of the protocols that was produced after careful reading of the RFC's and with some interaction with the Internet technical community, and that followed good communications software engineering practices, should differ from the requirements of this document in only minor ways. Thus, in many cases, the "requirements" in this RFC are already stated or implied in the standard protocol documents, so that their inclusion here is, in a sense, redundant. However, they were included because some past implementation has made the wrong choice, causing problems of interoperability, performance, and/or robustness.

This document includes discussion and explanation of many of the requirements and recommendations. A simple list of requirements would be dangerous, because:

- Some required features are more important than others, and some features are optional.

- o There may be valid reasons why particular vendor products that are designed for restricted contexts might choose to use different specifications.

However, the specifications of this document must be followed to meet the general goal of arbitrary host interoperation across the diversity and complexity of the Internet system. Although most current implementations fail to meet these requirements in various ways, some minor and some major, this specification is the ideal towards which we need to move.

These requirements are based on the current level of Internet architecture. This document will be updated as required to provide additional clarifications or to include additional information in those areas in which specifications are still evolving.

This introductory section begins with a brief overview of the Internet architecture as it relates to hosts, and then gives some general advice to host software vendors. Finally, there is some guidance on reading the rest of the document and some terminology.

1.1 The Internet Architecture

General background and discussion on the Internet architecture and supporting protocol suite can be found in the DDN Protocol Handbook [INTRO:3]; for background see for example [INTRO:9], [INTRO:10], and [INTRO:11]. Reference [INTRO:5] describes the procedure for obtaining Internet protocol documents, while [INTRO:6] contains a list of the numbers assigned within Internet protocols.

1.1.1 Internet Hosts

A host computer, or simply "host," is the ultimate consumer of communication services. A host generally executes application programs on behalf of user(s), employing network and/or Internet communication services in support of this function. An Internet host corresponds to the concept of an "End-System" used in the OSI protocol suite [INTRO:13].

An Internet communication system consists of interconnected packet networks supporting communication among host computers using the Internet protocols. The networks are interconnected using packet-switching computers called "gateways" or "IP routers" by the Internet community, and "Intermediate Systems" by the OSI world [INTRO:13]. The RFC "Requirements for Internet Gateways" [INTRO:2] contains the official specifications for Internet gateways. That RFC together with

the present document and its companion [INTRO:1] define the rules for the current realization of the Internet architecture.

Internet hosts span a wide range of size, speed, and function. They range in size from small microprocessors through workstations to mainframes and supercomputers. In function, they range from single-purpose hosts (such as terminal servers) to full-service hosts that support a variety of online network services, typically including remote login, file transfer, and electronic mail.

A host is generally said to be multihomed if it has more than one interface to the same or to different networks. See Section 1.1.3 on "Terminology".

1.1.2 Architectural Assumptions

The current Internet architecture is based on a set of assumptions about the communication system. The assumptions most relevant to hosts are as follows:

(a) The Internet is a network of networks.

Each host is directly connected to some particular network(s); its connection to the Internet is only conceptual. Two hosts on the same network communicate with each other using the same set of protocols that they would use to communicate with hosts on distant networks.

(b) Gateways don’t keep connection state information.

To improve robustness of the communication system, gateways are designed to be stateless, forwarding each IP datagram independently of other datagrams. As a result, redundant paths can be exploited to provide robust service in spite of failures of intervening gateways and networks.

All state information required for end-to-end flow control and reliability is implemented in the hosts, in the transport layer or in application programs. All connection control information is thus co-located with the end points of the communication, so it will be lost only if an end point fails.

(c) Routing complexity should be in the gateways.

Routing is a complex and difficult problem, and ought to be performed by the gateways, not the hosts. An important

objective is to insulate host software from changes caused by the inevitable evolution of the Internet routing architecture.

(d) The System must tolerate wide network variation.

A basic objective of the Internet design is to tolerate a wide range of network characteristics -- e.g., bandwidth, delay, packet loss, packet reordering, and maximum packet size. Another objective is robustness against failure of individual networks, gateways, and hosts, using whatever bandwidth is still available. Finally, the goal is full "open system interconnection": an Internet host must be able to interoperate robustly and effectively with any other Internet host, across diverse Internet paths.

Sometimes host implementors have designed for less ambitious goals. For example, the LAN environment is typically much more benign than the Internet as a whole; LANs have low packet loss and delay and do not reorder packets. Some vendors have fielded host implementations that are adequate for a simple LAN environment, but work badly for general interoperation. The vendor justifies such a product as being economical within the restricted LAN market. However, isolated LANs seldom stay isolated for long; they are soon gatewayed to each other, to organization-wide internets, and eventually to the global Internet system. In the end, neither the customer nor the vendor is served by incomplete or substandard Internet host software.

The requirements spelled out in this document are designed for a full-function Internet host, capable of full interoperation over an arbitrary Internet path.

1.1.3 Internet Protocol Suite

To communicate using the Internet system, a host must implement the layered set of protocols comprising the Internet protocol suite. A host typically must implement at least one protocol from each layer.

The protocol layers used in the Internet architecture are as follows [INTRO:4]:

- Application Layer

The application layer is the top layer of the Internet protocol suite. The Internet suite does not further subdivide the application layer, although some of the Internet application layer protocols do contain some internal sub-layering. The application layer of the Internet suite essentially combines the functions of the top two layers -- Presentation and Application -- of the OSI reference model.

We distinguish two categories of application layer protocols: user protocols that provide service directly to users, and support protocols that provide common system functions. Requirements for user and support protocols will be found in the companion RFC [INTRO:1].

The most common Internet user protocols are:

- Telnet (remote login)
- FTP (file transfer)
- SMTP (electronic mail delivery)

There are a number of other standardized user protocols [INTRO:4] and many private user protocols.

Support protocols, used for host name mapping, booting, and management, include SNMP, BOOTP, RARP, and the Domain Name System (DNS) protocols.

- Transport Layer

The transport layer provides end-to-end communication services for applications. There are two primary transport layer protocols at present:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

TCP is a reliable connection-oriented transport service that provides end-to-end reliability, resequencing, and flow control. UDP is a connectionless ("datagram") transport service.

Other transport protocols have been developed by the research community, and the set of official Internet transport protocols may be expanded in the future.

Transport layer protocols are discussed in Chapter 4.

- Internet Layer

  All Internet transport protocols use the Internet Protocol (IP) to carry data from source host to destination host. IP is a connectionless or datagram internetwork service, providing no end-to-end delivery guarantees. Thus, IP datagrams may arrive at the destination host damaged, duplicated, out of order, or not at all. The layers above IP are responsible for reliable delivery service when it is required. The IP protocol includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security information.

  The datagram or connectionless nature of the IP protocol is a fundamental and characteristic feature of the Internet architecture. Internet IP was the model for the OSI Connectionless Network Protocol [INTRO:12].

  ICMP is a control protocol that is considered to be an integral part of IP, although it is architecturally layered upon IP, i.e., it uses IP to carry its data end-to-end just as a transport protocol like TCP or UDP does. ICMP provides error reporting, congestion reporting, and first-hop gateway redirection.

  IGMP is an Internet layer protocol used for establishing dynamic host groups for IP multicasting.

  The Internet layer protocols IP, ICMP, and IGMP are discussed in Chapter 3.

- Link Layer

  To communicate on its directly-connected network, a host must implement the communication protocol used to interface to that network. We call this a link layer or media-access layer protocol.

  There is a wide variety of link layer protocols, corresponding to the many different types of networks. See Chapter 2.

1.1.4 Embedded Gateway Code

Some Internet host software includes embedded gateway functionality, so that these hosts can forward packets as a

gateway would, while still performing the application layer functions of a host.

Such dual-purpose systems must follow the Gateway Requirements RFC [INTRO:2] with respect to their gateway functions, and must follow the present document with respect to their host functions. In all overlapping cases, the two specifications should be in agreement.

There are varying opinions in the Internet community about embedded gateway functionality. The main arguments are as follows:

- Pro: in a local network environment where networking is informal, or in isolated internets, it may be convenient and economical to use existing host systems as gateways.

  There is also an architectural argument for embedded gateway functionality: multihoming is much more common than originally foreseen, and multihoming forces a host to make routing decisions as if it were a gateway. If the multihomed host contains an embedded gateway, it will have full routing knowledge and as a result will be able to make more optimal routing decisions.

- Con: Gateway algorithms and protocols are still changing, and they will continue to change as the Internet system grows larger. Attempting to include a general gateway function within the host IP layer will force host system maintainers to track these (more frequent) changes. Also, a larger pool of gateway implementations will make coordinating the changes more difficult. Finally, the complexity of a gateway IP layer is somewhat greater than that of a host, making the implementation and operation tasks more complex.

  In addition, the style of operation of some hosts is not appropriate for providing stable and robust gateway service.

There is considerable merit in both of these viewpoints. One conclusion can be drawn: an host administrator must have conscious control over whether or not a given host acts as a gateway. See Section 3.1 for the detailed requirements.

1.2 General Considerations

There are two important lessons that vendors of Internet host software have learned and which a new vendor should consider seriously.

1.2.1 Continuing Internet Evolution

The enormous growth of the Internet has revealed problems of management and scaling in a large datagram-based packet communication system. These problems are being addressed, and as a result there will be continuing evolution of the specifications described in this document. These changes will be carefully planned and controlled, since there is extensive participation in this planning by the vendors and by the organizations responsible for operations of the networks.

Development, evolution, and revision are characteristic of computer network protocols today, and this situation will persist for some years. A vendor who develops computer communication software for the Internet protocol suite (or any other protocol suite!) and then fails to maintain and update that software for changing specifications is going to leave a trail of unhappy customers. The Internet is a large communication network, and the users are in constant contact through it. Experience has shown that knowledge of deficiencies in vendor software propagates quickly through the Internet technical community.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability [IP:1]:

> "Be liberal in what you accept, and
> conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events;

mere human malice would never have taken so devious a course!

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field -- e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up. An undefined code might be logged (see below), but it must not cause a failure.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is "watch out for misbehaving hosts"; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

1.2.3 Error Logging

The Internet includes a great variety of host and gateway systems, each implementing many protocols and protocol layers, and some of these contain bugs and mis-features in their Internet protocol software. As a result of complexity, diversity, and distribution of function, the diagnosis of Internet problems is often very difficult.

Problem diagnosis will be aided if host implementations include a carefully designed facility for logging erroneous or "strange" protocol events. It is important to include as much diagnostic information as possible when an error is logged. In particular, it is often useful to record the header(s) of a packet that caused an error. However, care must be taken to ensure that error logging does not consume prohibitive amounts of resources or otherwise interfere with the operation of the host.

There is a tendency for abnormal but harmless protocol events to overflow error logging files; this can be avoided by using a "circular" log, or by enabling logging only while diagnosing a known failure. It may be useful to filter and count duplicate successive messages. One strategy that seems to work well is: (1) always count abnormalities and make such counts accessible through the management protocol (see [INTRO:1]); and (2) allow

the logging of a great variety of events to be selectively enabled. For example, it might useful to be able to "log everything" or to "log everything for host X".

Note that different managements may have differing policies about the amount of error logging that they want normally enabled in a host. Some will say, "if it doesn't hurt me, I don't want to know about it", while others will want to take a more watchful and aggressive attitude about detecting and removing protocol abnormalities.

1.2.4 Configuration

It would be ideal if a host implementation of the Internet protocol suite could be entirely self-configuring. This would allow the whole suite to be implemented in ROM or cast into silicon, it would simplify diskless workstations, and it would be an immense boon to harried LAN administrators as well as system vendors. We have not reached this ideal; in fact, we are not even close.

At many points in this document, you will find a requirement that a parameter be a configurable option. There are several different reasons behind such requirements. In a few cases, there is current uncertainty or disagreement about the best value, and it may be necessary to update the recommended value in the future. In other cases, the value really depends on external factors -- e.g., the size of the host and the distribution of its communication load, or the speeds and topology of nearby networks -- and self-tuning algorithms are unavailable and may be insufficient. In some cases, configurability is needed because of administrative requirements.

Finally, some configuration options are required to communicate with obsolete or incorrect implementations of the protocols, distributed without sources, that unfortunately persist in many parts of the Internet. To make correct systems coexist with these faulty systems, administrators often have to "mis-configure" the correct systems. This problem will correct itself gradually as the faulty systems are retired, but it cannot be ignored by vendors.

When we say that a parameter must be configurable, we do not intend to require that its value be explicitly read from a configuration file at every boot time. We recommend that implementors set up a default for each parameter, so a configuration file is only necessary to override those defaults

that are inappropriate in a particular installation. Thus, the configurability requirement is an assurance that it will be POSSIBLE to override the default when necessary, even in a binary-only or ROM-based product.

This document requires a particular value for such defaults in some cases. The choice of default is a sensitive issue when the configuration item controls the accommodation to existing faulty systems. If the Internet is to converge successfully to complete interoperability, the default values built into implementations must implement the official protocol, not "mis-configurations" to accommodate faulty implementations. Although marketing considerations have led some vendors to choose mis-configuration defaults, we urge vendors to choose defaults that will conform to the standard.

Finally, we note that a vendor needs to provide adequate documentation on all configuration parameters, their limits and effects.

1.3 Reading this Document

1.3.1 Organization

Protocol layering, which is generally used as an organizing principle in implementing network software, has also been used to organize this document. In describing the rules, we assume that an implementation does strictly mirror the layering of the protocols. Thus, the following three major sections specify the requirements for the link layer, the internet layer, and the transport layer, respectively. A companion RFC [INTRO:1] covers application level software. This layerist organization was chosen for simplicity and clarity.

However, strict layering is an imperfect model, both for the protocol suite and for recommended implementation approaches. Protocols in different layers interact in complex and sometimes subtle ways, and particular functions often involve multiple layers. There are many design choices in an implementation, many of which involve creative "breaking" of strict layering. Every implementor is urged to read references [INTRO:7] and [INTRO:8].

This document describes the conceptual service interface between layers using a functional ("procedure call") notation, like that used in the TCP specification [TCP:1]. A host implementation must support the logical information flow

implied by these calls, but need not literally implement the calls themselves. For example, many implementations reflect the coupling between the transport layer and the IP layer by giving them shared access to common data structures. These data structures, rather than explicit procedure calls, are then the agency for passing much of the information that is required.

In general, each major section of this document is organized into the following subsections:

(1) Introduction

(2) Protocol Walk-Through -- considers the protocol specification documents section-by-section, correcting errors, stating requirements that may be ambiguous or ill-defined, and providing further clarification or explanation.

(3) Specific Issues -- discusses protocol design and implementation issues that were not included in the walk-through.

(4) Interfaces -- discusses the service interface to the next higher layer.

(5) Summary -- contains a summary of the requirements of the section.

Under many of the individual topics in this document, there is parenthetical material labeled "DISCUSSION" or "IMPLEMENTATION". This material is intended to give clarification and explanation of the preceding requirements text. It also includes some suggestions on possible future directions or developments. The implementation material contains suggested approaches that an implementor may want to consider.

The summary sections are intended to be guides and indexes to the text, but are necessarily cryptic and incomplete. The summaries should never be used or referenced separately from the complete RFC.

1.3.2 Requirements

In this document, the words that are used to define the significance of each particular requirement are capitalized.

These words are:

* "MUST"

  This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

* "SHOULD"

  This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant".

1.3.3 Terminology

This document uses the following technical terms:

Segment
: A segment is the unit of end-to-end transmission in the TCP protocol. A segment consists of a TCP header followed by application data. A segment is transmitted by encapsulation inside an IP datagram.

Message
: In this description of the lower-layer protocols, a message is the unit of transmission in a transport layer protocol. In particular, a TCP segment is a message. A message consists of a transport protocol header followed by application protocol data. To be transmitted end-to-

end through the Internet, a message must be encapsulated inside a datagram.

IP Datagram
: An IP datagram is the unit of end-to-end transmission in the IP protocol. An IP datagram consists of an IP header followed by transport layer data, i.e., of an IP header followed by a message.

  In the description of the internet layer (Section 3), the unqualified term "datagram" should be understood to refer to an IP datagram.

Packet
: A packet is the unit of data passed across the interface between the internet layer and the link layer. It includes an IP header and data. A packet may be a complete IP datagram or a fragment of an IP datagram.

Frame
: A frame is the unit of transmission in a link layer protocol, and consists of a link-layer header followed by a packet.

Connected Network
: A network to which a host is interfaced is often known as the "local network" or the "subnetwork" relative to that host. However, these terms can cause confusion, and therefore we use the term "connected network" in this document.

Multihomed
: A host is said to be multihomed if it has multiple IP addresses. For a discussion of multihoming, see Section 3.3.4 below.

Physical network interface
: This is a physical interface to a connected network and has a (possibly unique) link-layer address. Multiple physical network interfaces on a single host may share the same link-layer address, but the address must be unique for different hosts on the same physical network.

Logical [network] interface
: We define a logical [network] interface to be a logical path, distinguished by a unique IP address, to a connected network. See Section 3.3.4.

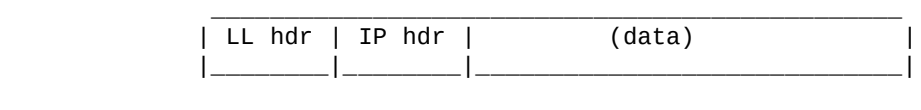
Specific-destination address
     This is the effective destination address of a datagram, even if it is broadcast or multicast; see Section 3.2.1.3.

Path
     At a given moment, all the IP datagrams from a particular source host to a particular destination host will typically traverse the same sequence of gateways. We use the term "path" for this sequence. Note that a path is uni-directional; it is not unusual to have different paths in the two directions between a given host pair.

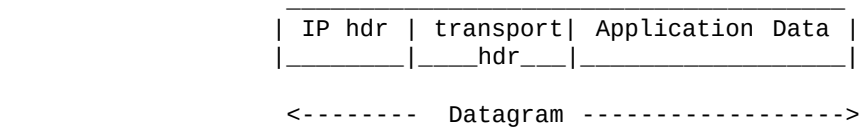
MTU
     The maximum transmission unit, i.e., the size of the largest packet that can be transmitted.

The terms frame, packet, datagram, message, and segment are illustrated by the following schematic diagrams:

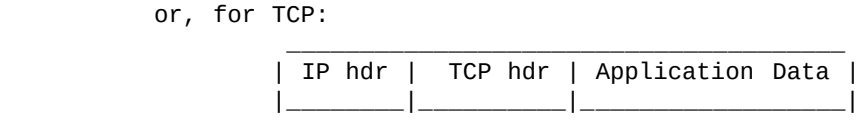
A. Transmission on connected network:



B. Before IP fragmentation or after IP reassembly:



or, for TCP:



1.4 Acknowledgments

This document incorporates contributions and comments from a large group of Internet protocol experts, including representatives of university and research labs, vendors, and government agencies. It was assembled primarily by the Host Requirements Working Group of the Internet Engineering Task Force (IETF).

The Editor would especially like to acknowledge the tireless dedication of the following people, who attended many long meetings and generated 3 million bytes of electronic mail over the past 18 months in pursuit of this document: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), and James Van Bokkelen (FTP Software).

In addition, the following people made major contributions to the effort: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), and Mike StJohns (DCA). The following also made significant contributions to particular areas: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), and Rayan Zachariassen (Toronto).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

2. LINK LAYER

2.1 INTRODUCTION

All Internet systems, both hosts and gateways, have the same requirements for link layer protocols. These requirements are given in Chapter 3 of "Requirements for Internet Gateways" [INTRO:2], augmented with the material in this section.

2.2 PROTOCOL WALK-THROUGH

None.

2.3 SPECIFIC ISSUES

2.3.1 Trailer Protocol Negotiation

The trailer protocol [LINK:1] for link-layer encapsulation MAY be used, but only when it has been verified that both systems (host or gateway) involved in the link-layer communication implement trailers. If the system does not dynamically negotiate use of the trailer protocol on a per-destination basis, the default configuration MUST disable the protocol.

DISCUSSION:

The trailer protocol is a link-layer encapsulation technique that rearranges the data contents of packets sent on the physical network. In some cases, trailers improve the throughput of higher layer protocols by reducing the amount of data copying within the operating system. Higher layer protocols are unaware of trailer use, but both the sending and receiving host MUST understand the protocol if it is used.

Improper use of trailers can result in very confusing symptoms. Only packets with specific size attributes are encapsulated using trailers, and typically only a small fraction of the packets being exchanged have these attributes. Thus, if a system using trailers exchanges packets with a system that does not, some packets disappear into a black hole while others are delivered successfully.

IMPLEMENTATION:

On an Ethernet, packets encapsulated with trailers use a distinct Ethernet type [LINK:1], and trailer negotiation is performed at the time that ARP is used to discover the link-layer address of a destination system.

Specifically, the ARP exchange is completed in the usual manner using the normal IP protocol type, but a host that wants to speak trailers will send an additional "trailer ARP reply" packet, i.e., an ARP reply that specifies the trailer encapsulation protocol type but otherwise has the format of a normal ARP reply. If a host configured to use trailers receives a trailer ARP reply message from a remote machine, it can add that machine to the list of machines that understand trailers, e.g., by marking the corresponding entry in the ARP cache.

Hosts wishing to receive trailer encapsulations send trailer ARP replies whenever they complete exchanges of normal ARP messages for IP. Thus, a host that received an ARP request for its IP protocol address would send a trailer ARP reply in addition to the normal IP ARP reply; a host that sent the IP ARP request would send a trailer ARP reply when it received the corresponding IP ARP reply. In this way, either the requesting or responding host in an IP ARP exchange may request that it receive trailer encapsulations.

This scheme, using extra trailer ARP reply packets rather than sending an ARP request for the trailer protocol type, was designed to avoid a continuous exchange of ARP packets with a misbehaving host that, contrary to any specification or common sense, responded to an ARP reply for trailers with another ARP reply for IP. This problem is avoided by sending a trailer ARP reply in response to an IP ARP reply only when the IP ARP reply answers an outstanding request; this is true when the hardware address for the host is still unknown when the IP ARP reply is received. A trailer ARP reply may always be sent along with an IP ARP reply responding to an IP ARP request.

2.3.2 Address Resolution Protocol -- ARP

2.3.2.1 ARP Cache Validation

An implementation of the Address Resolution Protocol (ARP) [LINK:2] MUST provide a mechanism to flush out-of-date cache entries. If this mechanism involves a timeout, it SHOULD be possible to configure the timeout value.

A mechanism to prevent ARP flooding (repeatedly sending an ARP Request for the same IP address, at a high rate) MUST be included. The recommended maximum rate is 1 per second per

destination.

DISCUSSION:

The ARP specification [LINK:2] suggests but does not require a timeout mechanism to invalidate cache entries when hosts change their Ethernet addresses. The prevalence of proxy ARP (see Section 2.4 of [INTRO:2]) has significantly increased the likelihood that cache entries in hosts will become invalid, and therefore some ARP-cache invalidation mechanism is now required for hosts. Even in the absence of proxy ARP, a long-period cache timeout is useful in order to automatically correct any bad ARP data that might have been cached.

IMPLEMENTATION:

Four mechanisms have been used, sometimes in combination, to flush out-of-date cache entries.

(1) Timeout -- Periodically time out cache entries, even if they are in use. Note that this timeout should be restarted when the cache entry is "refreshed" (by observing the source fields, regardless of target address, of an ARP broadcast from the system in question). For proxy ARP situations, the timeout needs to be on the order of a minute.

(2) Unicast Poll -- Actively poll the remote host by periodically sending a point-to-point ARP Request to it, and delete the entry if no ARP Reply is received from N successive polls. Again, the timeout should be on the order of a minute, and typically N is 2.

(3) Link-Layer Advice -- If the link-layer driver detects a delivery problem, flush the corresponding ARP cache entry.

(4) Higher-layer Advice -- Provide a call from the Internet layer to the link layer to indicate a delivery problem. The effect of this call would be to invalidate the corresponding cache entry. This call would be analogous to the "ADVISE_DELIVPROB()" call from the transport layer to the Internet layer (see Section 3.4), and in fact the ADVISE_DELIVPROB routine might in turn call the link-layer advice routine to invalidate

the ARP cache entry.

Approaches (1) and (2) involve ARP cache timeouts on the order of a minute or less. In the absence of proxy ARP, a timeout this short could create noticeable overhead traffic on a very large Ethernet. Therefore, it may be necessary to configure a host to lengthen the ARP cache timeout.

2.3.2.2 ARP Packet Queue

The link layer SHOULD save (rather than discard) at least one (the latest) packet of each set of packets destined to the same unresolved IP address, and transmit the saved packet when the address has been resolved.

DISCUSSION:

Failure to follow this recommendation causes the first packet of every exchange to be lost. Although higher-layer protocols can generally cope with packet loss by retransmission, packet loss does impact performance. For example, loss of a TCP open request causes the initial round-trip time estimate to be inflated. UDP-based applications such as the Domain Name System are more seriously affected.

2.3.3 Ethernet and IEEE 802 Encapsulation

The IP encapsulation for Ethernets is described in RFC-894 [LINK:3], while RFC-1042 [LINK:4] describes the IP encapsulation for IEEE 802 networks. RFC-1042 elaborates and replaces the discussion in Section 3.4 of [INTRO:2].

Every Internet host connected to a 10Mbps Ethernet cable:

- MUST be able to send and receive packets using RFC-894 encapsulation;
- SHOULD be able to receive RFC-1042 packets, intermixed with RFC-894 packets; and
- MAY be able to send packets using RFC-1042 encapsulation.

An Internet host that implements sending both the RFC-894 and the RFC-1042 encapsulations MUST provide a configuration switch to select which is sent, and this switch MUST default to RFC-894.

Note that the standard IP encapsulation in RFC-1042 does not use the protocol id value (K1=6) that IEEE reserved for IP; instead, it uses a value (K1=170) that implies an extension (the "SNAP") which can be used to hold the Ether-Type field. An Internet system MUST NOT send 802 packets using K1=6.

Address translation from Internet addresses to link-layer addresses on Ethernet and IEEE 802 networks MUST be managed by the Address Resolution Protocol (ARP).

The MTU for an Ethernet is 1500 and for 802.3 is 1492.

DISCUSSION:

The IEEE 802.3 specification provides for operation over a 10Mbps Ethernet cable, in which case Ethernet and IEEE 802.3 frames can be physically intermixed. A receiver can distinguish Ethernet and 802.3 frames by the value of the 802.3 Length field; this two-octet field coincides in the header with the Ether-Type field of an Ethernet frame. In particular, the 802.3 Length field must be less than or equal to 1500, while all valid Ether-Type values are greater than 1500.

Another compatibility problem arises with link-layer broadcasts. A broadcast sent with one framing will not be seen by hosts that can receive only the other framing.

The provisions of this section were designed to provide direct interoperation between 894-capable and 1042-capable systems on the same cable, to the maximum extent possible. It is intended to support the present situation where 894-only systems predominate, while providing an easy transition to a possible future in which 1042-capable systems become common.

Note that 894-only systems cannot interoperate directly with 1042-only systems. If the two system types are set up as two different logical networks on the same cable, they can communicate only through an IP gateway. Furthermore, it is not useful or even possible for a dual-format host to discover automatically which format to send, because of the problem of link-layer broadcasts.

2.4 LINK/INTERNET LAYER INTERFACE

The packet receive interface between the IP layer and the link layer MUST include a flag to indicate whether the incoming packet was addressed to a link-layer broadcast address.

DISCUSSION

Although the IP layer does not generally know link layer addresses (since every different network medium typically has a different address format), the broadcast address on a broadcast-capable medium is an important special case. See Section 3.2.2, especially the DISCUSSION concerning broadcast storms.

The packet send interface between the IP and link layers MUST include the 5-bit TOS field (see Section 3.2.1.6).

The link layer MUST NOT report a Destination Unreachable error to IP solely because there is no ARP cache entry for a destination.

2.5 LINK LAYER REQUIREMENTS SUMMARY

FEATURE	SECTION				SHOULD	MUST	Footnote
Trailer encapsulation	2.3.1			x			
Send Trailers by default without negotiation	2.3.1					x	
ARP	2.3.2						
Flush out-of-date ARP cache entries	2.3.2.1	x					
Prevent ARP floods	2.3.2.1	x					
Cache timeout configurable	2.3.2.1		x				
Save at least one (latest) unresolved pkt	2.3.2.2		x				
Ethernet and IEEE 802 Encapsulation	2.3.3						
Host able to:	2.3.3						
Send & receive RFC-894 encapsulation	2.3.3	x					
Receive RFC-1042 encapsulation	2.3.3		x				
Send RFC-1042 encapsulation	2.3.3			x			
Then config. sw. to select, RFC-894 dflt	2.3.3	x					
Send K1=6 encapsulation	2.3.3					x	
Use ARP on Ethernet and IEEE 802 nets	2.3.3	x					
Link layer report b'casts to IP layer	2.4	x					
IP layer pass TOS to link layer	2.4	x					
No ARP cache entry treated as Dest. Unreach.	2.4					x	

3. INTERNET LAYER PROTOCOLS

3.1 INTRODUCTION

The Robustness Principle: "Be liberal in what you accept, and conservative in what you send" is particularly important in the Internet layer, where one misbehaving host can deny Internet service to many other hosts.

The protocol standards used in the Internet layer are:

- RFC-791 [IP:1] defines the IP protocol and gives an introduction to the architecture of the Internet.

- RFC-792 [IP:2] defines ICMP, which provides routing, diagnostic and error functionality for IP. Although ICMP messages are encapsulated within IP datagrams, ICMP processing is considered to be (and is typically implemented as) part of the IP layer. See Section 3.2.2.

- RFC-950 [IP:3] defines the mandatory subnet extension to the addressing architecture.

- RFC-1112 [IP:4] defines the Internet Group Management Protocol IGMP, as part of a recommended extension to hosts and to the host-gateway interface to support Internet-wide multicasting at the IP level. See Section 3.2.3.

  The target of an IP multicast may be an arbitrary group of Internet hosts. IP multicasting is designed as a natural extension of the link-layer multicasting facilities of some networks, and it provides a standard means for local access to such link-layer multicasting facilities.

Other important references are listed in Section 5 of this document.

The Internet layer of host software MUST implement both IP and ICMP. See Section 3.3.7 for the requirements on support of IGMP.

The host IP layer has two basic functions: (1) choose the "next hop" gateway or host for outgoing IP datagrams and (2) reassemble incoming IP datagrams. The IP layer may also (3) implement intentional fragmentation of outgoing datagrams. Finally, the IP layer must (4) provide diagnostic and error functionality. We expect that IP layer functions may increase somewhat in the future, as further Internet control and management facilities are developed.

For normal datagrams, the processing is straightforward. For incoming datagrams, the IP layer:

(1) verifies that the datagram is correctly formatted;

(2) verifies that it is destined to the local host;

(3) processes options;

(4) reassembles the datagram if necessary; and

(5) passes the encapsulated message to the appropriate transport-layer protocol module.

For outgoing datagrams, the IP layer:

(1) sets any fields not set by the transport layer;

(2) selects the correct first hop on the connected network (a process called "routing");

(3) fragments the datagram if necessary and if intentional fragmentation is implemented (see Section 3.3.3); and

(4) passes the packet(s) to the appropriate link-layer driver.

A host is said to be multihomed if it has multiple IP addresses. Multihoming introduces considerable confusion and complexity into the protocol suite, and it is an area in which the Internet architecture falls seriously short of solving all problems. There are two distinct problem areas in multihoming:

(1) Local multihoming -- the host itself is multihomed; or

(2) Remote multihoming -- the local host needs to communicate with a remote multihomed host.

At present, remote multihoming MUST be handled at the application layer, as discussed in the companion RFC [INTRO:1]. A host MAY support local multihoming, which is discussed in this document, and in particular in Section 3.3.4.

Any host that forwards datagrams generated by another host is acting as a gateway and MUST also meet the specifications laid out in the gateway requirements RFC [INTRO:2]. An Internet host that includes embedded gateway code MUST have a configuration switch to disable the gateway function, and this switch MUST default to the

non-gateway mode. In this mode, a datagram arriving through one interface will not be forwarded to another host or gateway (unless it is source-routed), regardless of whether the host is single-homed or multihomed. The host software MUST NOT automatically move into gateway mode if the host has more than one interface, as the operator of the machine may neither want to provide that service nor be competent to do so.

In the following, the action specified in certain cases is to "silently discard" a received datagram. This means that the datagram will be discarded without further processing and that the host will not send any ICMP error message (see Section 3.2.2) as a result. However, for diagnosis of problems a host SHOULD provide the capability of logging the error (see Section 1.2.3), including the contents of the silently-discarded datagram, and SHOULD record the event in a statistics counter.

DISCUSSION:
Silent discard of erroneous datagrams is generally intended to prevent "broadcast storms".

3.2 PROTOCOL WALK-THROUGH

3.2.1 Internet Protocol -- IP

3.2.1.1 Version Number: RFC-791 Section 3.1

A datagram whose version number is not 4 MUST be silently discarded.

3.2.1.2 Checksum: RFC-791 Section 3.1

A host MUST verify the IP header checksum on every received datagram and silently discard every datagram that has a bad checksum.

3.2.1.3 Addressing: RFC-791 Section 3.2

There are now five classes of IP addresses: Class A through Class E. Class D addresses are used for IP multicasting [IP:4], while Class E addresses are reserved for experimental use.

A multicast (Class D) address is a 28-bit logical address that stands for a group of hosts, and may be either permanent or transient. Permanent multicast addresses are allocated by the Internet Assigned Number Authority [INTRO:6], while transient addresses may be allocated

dynamically to transient groups. Group membership is determined dynamically using IGMP [IP:4].

We now summarize the important special cases for Class A, B, and C IP addresses, using the following notation for an IP address:

{ <Network-number>, <Host-number> }

or

{ <Network-number>, <Subnet-number>, <Host-number> }

and the notation "-1" for a field that contains all 1 bits. This notation is not intended to imply that the 1-bits in an address mask need be contiguous.

(a) { 0, 0 }

This host on this network. MUST NOT be sent, except as a source address as part of an initialization procedure by which the host learns its own IP address.

See also Section 3.3.6 for a non-standard use of {0,0}.

(b) { 0, <Host-number> }

Specified host on this network. It MUST NOT be sent, except as a source address as part of an initialization procedure by which the host learns its full IP address.

(c) { -1, -1 }

Limited broadcast. It MUST NOT be used as a source address.

A datagram with this destination address will be received by every host on the connected physical network but will not be forwarded outside that network.

(d) { <Network-number>, -1 }

Directed broadcast to the specified network. It MUST NOT be used as a source address.

(e) { <Network-number>, <Subnet-number>, -1 }

Directed broadcast to the specified subnet. It MUST NOT be used as a source address.

(f) { <Network-number>, -1, -1 }

Directed broadcast to all subnets of the specified subnetted network. It MUST NOT be used as a source address.

(g) { 127, <any> }

Internal host loopback address. Addresses of this form MUST NOT appear outside a host.

The <Network-number> is administratively assigned so that its value will be unique in the entire world.

IP addresses are not permitted to have the value 0 or -1 for any of the <Host-number>, <Network-number>, or <Subnet-number> fields (except in the special cases listed above). This implies that each of these fields will be at least two bits long.

For further discussion of broadcast addresses, see Section 3.3.6.

A host MUST support the subnet extensions to IP [IP:3]. As a result, there will be an address mask of the form: {-1, -1, 0} associated with each of the host's local IP addresses; see Sections 3.2.2.9 and 3.3.1.1.

When a host sends any datagram, the IP source address MUST be one of its own IP addresses (but not a broadcast or multicast address).

A host MUST silently discard an incoming datagram that is not destined for the host. An incoming datagram is destined for the host if the datagram's destination address field is:

(1) (one of) the host's IP address(es); or

(2) an IP broadcast address valid for the connected network; or

(3) the address for a multicast group of which the host is a member on the incoming physical interface.

For most purposes, a datagram addressed to a broadcast or multicast destination is processed as if it had been addressed to one of the host's IP addresses; we use the term "specific-destination address" for the equivalent local IP

address of the host. The specific-destination address is defined to be the destination address in the IP header unless the header contains a broadcast or multicast address, in which case the specific-destination is an IP address assigned to the physical interface on which the datagram arrived.

A host MUST silently discard an incoming datagram containing an IP source address that is invalid by the rules of this section. This validation could be done in either the IP layer or by each protocol in the transport layer.

DISCUSSION:

A mis-addressed datagram might be caused by a link-layer broadcast of a unicast datagram or by a gateway or host that is confused or mis-configured.

An architectural goal for Internet hosts was to allow IP addresses to be featureless 32-bit numbers, avoiding algorithms that required a knowledge of the IP address format. Otherwise, any future change in the format or interpretation of IP addresses will require host software changes. However, validation of broadcast and multicast addresses violates this goal; a few other violations are described elsewhere in this document.

Implementers should be aware that applications depending upon the all-subnets directed broadcast address (f) may be unusable on some networks. All-subnets broadcast is not widely implemented in vendor gateways at present, and even when it is implemented, a particular network administration may disable it in the gateway configuration.

3.2.1.4 Fragmentation and Reassembly: RFC-791 Section 3.2

The Internet model requires that every host support reassembly. See Sections 3.3.2 and 3.3.3 for the requirements on fragmentation and reassembly.

3.2.1.5 Identification: RFC-791 Section 3.2

When sending an identical copy of an earlier datagram, a host MAY optionally retain the same Identification field in the copy.

DISCUSSION:

Some Internet protocol experts have maintained that when a host sends an identical copy of an earlier datagram, the new copy should contain the same Identification value as the original. There are two suggested advantages: (1) if the datagrams are fragmented and some of the fragments are lost, the receiver may be able to reconstruct a complete datagram from fragments of the original and the copies; (2) a congested gateway might use the IP Identification field (and Fragment Offset) to discard duplicate datagrams from the queue.

However, the observed patterns of datagram loss in the Internet do not favor the probability of retransmitted fragments filling reassembly gaps, while other mechanisms (e.g., TCP repacketizing upon retransmission) tend to prevent retransmission of an identical datagram [IP:9]. Therefore, we believe that retransmitting the same Identification field is not useful. Also, a connectionless transport protocol like UDP would require the cooperation of the application programs to retain the same Identification value in identical datagrams.

3.2.1.6 Type-of-Service: RFC-791 Section 3.2

The "Type-of-Service" byte in the IP header is divided into two sections: the Precedence field (high-order 3 bits), and a field that is customarily called "Type-of-Service" or "TOS" (low-order 5 bits). In this document, all references to "TOS" or the "TOS field" refer to the low-order 5 bits only.

The Precedence field is intended for Department of Defense applications of the Internet protocols. The use of non-zero values in this field is outside the scope of this document and the IP standard specification. Vendors should consult the Defense Communication Agency (DCA) for guidance on the IP Precedence field and its implications for other protocol layers. However, vendors should note that the use of precedence will most likely require that its value be passed between protocol layers in just the same way as the TOS field is passed.

The IP layer MUST provide a means for the transport layer to set the TOS field of every datagram that is sent; the default is all zero bits. The IP layer SHOULD pass received

TOS values up to the transport layer.

The particular link-layer mappings of TOS contained in RFC-795 SHOULD NOT be implemented.

DISCUSSION:

While the TOS field has been little used in the past, it is expected to play an increasing role in the near future. The TOS field is expected to be used to control two aspects of gateway operations: routing and queueing algorithms. See Section 2 of [INTRO:1] for the requirements on application programs to specify TOS values.

The TOS field may also be mapped into link-layer service selectors. This has been applied to provide effective sharing of serial lines by different classes of TCP traffic, for example. However, the mappings suggested in RFC-795 for networks that were included in the Internet as of 1981 are now obsolete.

3.2.1.7 Time-to-Live: RFC-791 Section 3.2

A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero.

A host MUST NOT discard a datagram just because it was received with TTL less than 2.

The IP layer MUST provide a means for the transport layer to set the TTL field of every datagram that is sent. When a fixed TTL value is used, it MUST be configurable. The current suggested value will be published in the "Assigned Numbers" RFC.

DISCUSSION:

The TTL field has two functions: limit the lifetime of TCP segments (see RFC-793 [TCP:1], p. 28), and terminate Internet routing loops. Although TTL is a time in seconds, it also has some attributes of a hop-count, since each gateway is required to reduce the TTL field by at least one.

The intent is that TTL expiration will cause a datagram to be discarded by a gateway but not by the destination host; however, hosts that act as gateways by forwarding datagrams must follow the gateway rules for TTL.

A higher-layer protocol may want to set the TTL in order to implement an "expanding scope" search for some Internet resource. This is used by some diagnostic tools, and is expected to be useful for locating the "nearest" server of a given class using IP multicasting, for example. A particular transport protocol may also want to specify its own TTL bound on maximum datagram lifetime.

A fixed value must be at least big enough for the Internet "diameter," i.e., the longest possible path. A reasonable value is about twice the diameter, to allow for continued Internet growth.

3.2.1.8 Options: RFC-791 Section 3.2

There MUST be a means for the transport layer to specify IP options to be included in transmitted IP datagrams (see Section 3.4).

All IP options (except NOP or END-OF-LIST) received in datagrams MUST be passed to the transport layer (or to ICMP processing when the datagram is an ICMP message). The IP and transport layer MUST each interpret those IP options that they understand and silently ignore the others.

Later sections of this document discuss specific IP option support required by each of ICMP, TCP, and UDP.

DISCUSSION:

Passing all received IP options to the transport layer is a deliberate "violation of strict layering" that is designed to ease the introduction of new transport-relevant IP options in the future. Each layer must pick out any options that are relevant to its own processing and ignore the rest. For this purpose, every IP option except NOP and END-OF-LIST will include a specification of its own length.

This document does not define the order in which a receiver must process multiple options in the same IP header. Hosts sending multiple options must be aware that this introduces an ambiguity in the meaning of certain options when combined with a source-route option.

IMPLEMENTATION:

The IP layer must not crash as the result of an option

length that is outside the possible range. For example, erroneous option lengths have been observed to put some IP implementations into infinite loops.

Here are the requirements for specific IP options:

(a) Security Option

Some environments require the Security option in every datagram; such a requirement is outside the scope of this document and the IP standard specification. Note, however, that the security options described in RFC-791 and RFC-1038 are obsolete. For DoD applications, vendors should consult [IP:8] for guidance.

(b) Stream Identifier Option

This option is obsolete; it SHOULD NOT be sent, and it MUST be silently ignored if received.

(c) Source Route Options

A host MUST support originating a source route and MUST be able to act as the final destination of a source route.

If host receives a datagram containing a completed source route (i.e., the pointer points beyond the last field), the datagram has reached its final destination; the option as received (the recorded route) MUST be passed up to the transport layer (or to ICMP message processing). This recorded route will be reversed and used to form a return source route for reply datagrams (see discussion of IP Options in Section 4). When a return source route is built, it MUST be correctly formed even if the recorded route included the source host (see case (B) in the discussion below).

An IP header containing more than one Source Route option MUST NOT be sent; the effect on routing of multiple Source Route options is implementation-specific.

Section 3.3.5 presents the rules for a host acting as an intermediate hop in a source route, i.e., forwarding

a source-routed datagram.

DISCUSSION:

If a source-routed datagram is fragmented, each fragment will contain a copy of the source route. Since the processing of IP options (including a source route) must precede reassembly, the original datagram will not be reassembled until the final destination is reached.

Suppose a source routed datagram is to be routed from host S to host D via gateways G1, G2, ... Gn. There was an ambiguity in the specification over whether the source route option in a datagram sent out by S should be (A) or (B):

(A): {>>G2, G3, ... Gn, D} <--- CORRECT

(B): {S, >>G2, G3, ... Gn, D} <---- WRONG

(where >> represents the pointer). If (A) is sent, the datagram received at D will contain the option: {G1, G2, ... Gn >>}, with S and D as the IP source and destination addresses. If (B) were sent, the datagram received at D would again contain S and D as the same IP source and destination addresses, but the option would be: {S, G1, ...Gn >>}; i.e., the originating host would be the first hop in the route.

(d) Record Route Option

Implementation of originating and processing the Record Route option is OPTIONAL.

(e) Timestamp Option

Implementation of originating and processing the Timestamp option is OPTIONAL. If it is implemented, the following rules apply:

- The originating host MUST record a timestamp in a Timestamp option whose Internet address fields are not pre-specified or whose first pre-specified address is the host's interface address.

- The destination host MUST (if possible) add the current timestamp to a Timestamp option before passing the option to the transport layer or to ICMP for processing.

- A timestamp value MUST follow the rules given in Section 3.2.2.8 for the ICMP Timestamp message.

3.2.2 Internet Control Message Protocol -- ICMP

ICMP messages are grouped into two classes.

*
ICMP error messages:

Destination Unreachable	(see Section 3.2.2.1)
Redirect	(see Section 3.2.2.2)
Source Quench	(see Section 3.2.2.3)
Time Exceeded	(see Section 3.2.2.4)
Parameter Problem	(see Section 3.2.2.5)

*
ICMP query messages:

Echo	(see Section 3.2.2.6)
Information	(see Section 3.2.2.7)
Timestamp	(see Section 3.2.2.8)
Address Mask	(see Section 3.2.2.9)

If an ICMP message of unknown type is received, it MUST be silently discarded.

Every ICMP error message includes the Internet header and at least the first 8 data octets of the datagram that triggered the error; more than 8 octets MAY be sent; this header and data MUST be unchanged from the received datagram.

In those cases where the Internet layer is required to pass an ICMP error message to the transport layer, the IP protocol number MUST be extracted from the original header and used to select the appropriate transport protocol entity to handle the error.

An ICMP error message SHOULD be sent with normal (i.e., zero) TOS bits.

An ICMP error message MUST NOT be sent as the result of receiving:

* an ICMP error message, or

* a datagram destined to an IP broadcast or IP multicast address, or

* a datagram sent as a link-layer broadcast, or

* a non-initial fragment, or

* a datagram whose source address does not define a single host -- e.g., a zero address, a loopback address, a broadcast address, a multicast address, or a Class E address.

NOTE: THESE RESTRICTIONS TAKE PRECEDENCE OVER ANY REQUIREMENT ELSEWHERE IN THIS DOCUMENT FOR SENDING ICMP ERROR MESSAGES.

DISCUSSION:

These rules will prevent the "broadcast storms" that have resulted from hosts returning ICMP error messages in response to broadcast datagrams. For example, a broadcast UDP segment to a non-existent port could trigger a flood of ICMP Destination Unreachable datagrams from all machines that do not have a client for that destination port. On a large Ethernet, the resulting collisions can render the network useless for a second or more.

Every datagram that is broadcast on the connected network should have a valid IP broadcast address as its IP destination (see Section 3.3.6). However, some hosts violate this rule. To be certain to detect broadcast datagrams, therefore, hosts are required to check for a link-layer broadcast as well as an IP-layer broadcast address.

IMPLEMENTATION:

This requires that the link layer inform the IP layer when a link-layer broadcast datagram has been received; see Section 2.4.

3.2.2.1 Destination Unreachable: RFC-792

The following additional codes are hereby defined:

6 = destination network unknown

7 = destination host unknown

8 = source host isolated

9 = communication with destination network administratively prohibited

10 = communication with destination host administratively prohibited

11 = network unreachable for type of service

12 = host unreachable for type of service

A host SHOULD generate Destination Unreachable messages with code:

2 (Protocol Unreachable), when the designated transport protocol is not supported; or

3 (Port Unreachable), when the designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender.

A Destination Unreachable message that is received MUST be reported to the transport layer. The transport layer SHOULD use the information appropriately; for example, see Sections 4.1.3.3, 4.2.3.9, and 4.2.4 below. A transport protocol that has its own mechanism for notifying the sender that a port is unreachable (e.g., TCP, which sends RST segments) MUST nevertheless accept an ICMP Port Unreachable for the same purpose.

A Destination Unreachable message that is received with code 0 (Net), 1 (Host), or 5 (Bad Source Route) may result from a routing transient and MUST therefore be interpreted as only a hint, not proof, that the specified destination is unreachable [IP:11]. For example, it MUST NOT be used as proof of a dead gateway (see Section 3.3.1).

3.2.2.2 Redirect: RFC-792

A host SHOULD NOT send an ICMP Redirect message; Redirects are to be sent only by gateways.

A host receiving a Redirect message MUST update its routing information accordingly. Every host MUST be prepared to

accept both Host and Network Redirects and to process them as described in Section 3.3.1.2 below.

A Redirect message SHOULD be silently discarded if the new gateway address it specifies is not on the same connected (sub-) net through which the Redirect arrived [INTRO:2, Appendix A], or if the source of the Redirect is not the current first-hop gateway for the specified destination (see Section 3.3.1).

3.2.2.3 Source Quench: RFC-792

A host MAY send a Source Quench message if it is approaching, or has reached, the point at which it is forced to discard incoming datagrams due to a shortage of reassembly buffers or other resources. See Section 2.2.3 of [INTRO:2] for suggestions on when to send Source Quench.

If a Source Quench message is received, the IP layer MUST report it to the transport layer (or ICMP processing). In general, the transport or application layer SHOULD implement a mechanism to respond to Source Quench for any protocol that can send a sequence of datagrams to the same destination and which can reasonably be expected to maintain enough state information to make this feasible. See Section 4 for the handling of Source Quench by TCP and UDP.

DISCUSSION:

A Source Quench may be generated by the target host or by some gateway in the path of a datagram. The host receiving a Source Quench should throttle itself back for a period of time, then gradually increase the transmission rate again. The mechanism to respond to Source Quench may be in the transport layer (for connection-oriented protocols like TCP) or in the application layer (for protocols that are built on top of UDP).

A mechanism has been proposed [IP:14] to make the IP layer respond directly to Source Quench by controlling the rate at which datagrams are sent, however, this proposal is currently experimental and not currently recommended.

3.2.2.4 Time Exceeded: RFC-792

An incoming Time Exceeded message MUST be passed to the transport layer.

DISCUSSION:

A gateway will send a Time Exceeded Code 0 (In Transit) message when it discards a datagram due to an expired TTL field. This indicates either a gateway routing loop or too small an initial TTL value.

A host may receive a Time Exceeded Code 1 (Reassembly Timeout) message from a destination host that has timed out and discarded an incomplete datagram; see Section 3.3.2 below. In the future, receipt of this message might be part of some "MTU discovery" procedure, to discover the maximum datagram size that can be sent on the path without fragmentation.

3.2.2.5 Parameter Problem: RFC-792

A host SHOULD generate Parameter Problem messages. An incoming Parameter Problem message MUST be passed to the transport layer, and it MAY be reported to the user.

DISCUSSION:

The ICMP Parameter Problem message is sent to the source host for any problem not specifically covered by another ICMP message. Receipt of a Parameter Problem message generally indicates some local or remote implementation error.

A new variant on the Parameter Problem message is hereby defined:

Code 1 = required option is missing.

DISCUSSION:

This variant is currently in use in the military community for a missing security option.

3.2.2.6 Echo Request/Reply: RFC-792

Every host MUST implement an ICMP Echo server function that receives Echo Requests and sends corresponding Echo Replies. A host SHOULD also implement an application-layer interface for sending an Echo Request and receiving an Echo Reply, for diagnostic purposes.

An ICMP Echo Request destined to an IP broadcast or IP multicast address MAY be silently discarded.

DISCUSSION:

This neutral provision results from a passionate debate between those who feel that ICMP Echo to a broadcast address provides a valuable diagnostic capability and those who feel that misuse of this feature can too easily create packet storms.

The IP source address in an ICMP Echo Reply MUST be the same as the specific-destination address (defined in Section 3.2.1.3) of the corresponding ICMP Echo Request message.

Data received in an ICMP Echo Request MUST be entirely included in the resulting Echo Reply. However, if sending the Echo Reply requires intentional fragmentation that is not implemented, the datagram MUST be truncated to maximum transmission size (see Section 3.3.3) and sent.

Echo Reply messages MUST be passed to the ICMP user interface, unless the corresponding Echo Request originated in the IP layer.

If a Record Route and/or Time Stamp option is received in an ICMP Echo Request, this option (these options) SHOULD be updated to include the current host and included in the IP header of the Echo Reply message, without "truncation". Thus, the recorded route will be for the entire round trip.

If a Source Route option is received in an ICMP Echo Request, the return route MUST be reversed and used as a Source Route option for the Echo Reply message.

3.2.2.7 Information Request/Reply: RFC-792

A host SHOULD NOT implement these messages.

DISCUSSION:

The Information Request/Reply pair was intended to support self-configuring systems such as diskless workstations, to allow them to discover their IP network numbers at boot time. However, the RARP and BOOTP protocols provide better mechanisms for a host to discover its own IP address.

3.2.2.8 Timestamp and Timestamp Reply: RFC-792

A host MAY implement Timestamp and Timestamp Reply. If they are implemented, the following rules MUST be followed.

- The ICMP Timestamp server function returns a Timestamp Reply to every Timestamp message that is received. If this function is implemented, it SHOULD be designed for minimum variability in delay (e.g., implemented in the kernel to avoid delay in scheduling a user process).

The following cases for Timestamp are to be handled according to the corresponding rules for ICMP Echo:

- An ICMP Timestamp Request message to an IP broadcast or IP multicast address MAY be silently discarded.

- The IP source address in an ICMP Timestamp Reply MUST be the same as the specific-destination address of the corresponding Timestamp Request message.

- If a Source-route option is received in an ICMP Echo Request, the return route MUST be reversed and used as a Source Route option for the Timestamp Reply message.

- If a Record Route and/or Timestamp option is received in a Timestamp Request, this (these) option(s) SHOULD be updated to include the current host and included in the IP header of the Timestamp Reply message.

- Incoming Timestamp Reply messages MUST be passed up to the ICMP user interface.

The preferred form for a timestamp value (the "standard value") is in units of milliseconds since midnight Universal Time. However, it may be difficult to provide this value with millisecond resolution. For example, many systems use clocks that update only at line frequency, 50 or 60 times per second. Therefore, some latitude is allowed in a "standard value":

(a) A "standard value" MUST be updated at least 15 times per second (i.e., at most the six low-order bits of the value may be undefined).

(b) The accuracy of a "standard value" MUST approximate that of operator-set CPU clocks, i.e., correct within a few minutes.

3.2.2.9 Address Mask Request/Reply: RFC-950

A host MUST support the first, and MAY implement all three, of the following methods for determining the address mask(s) corresponding to its IP address(es):

(1) static configuration information;

(2) obtaining the address mask(s) dynamically as a side-effect of the system initialization process (see [INTRO:1]); and

(3) sending ICMP Address Mask Request(s) and receiving ICMP Address Mask Reply(s).

The choice of method to be used in a particular host MUST be configurable.

When method (3), the use of Address Mask messages, is enabled, then:

(a) When it initializes, the host MUST broadcast an Address Mask Request message on the connected network corresponding to the IP address. It MUST retransmit this message a small number of times if it does not receive an immediate Address Mask Reply.

(b) Until it has received an Address Mask Reply, the host SHOULD assume a mask appropriate for the address class of the IP address, i.e., assume that the connected network is not subnetted.

(c) The first Address Mask Reply message received MUST be used to set the address mask corresponding to the particular local IP address. This is true even if the first Address Mask Reply message is "unsolicited", in which case it will have been broadcast and may arrive after the host has ceased to retransmit Address Mask Requests. Once the mask has been set by an Address Mask Reply, later Address Mask Reply messages MUST be (silently) ignored.

Conversely, if Address Mask messages are disabled, then no ICMP Address Mask Requests will be sent, and any ICMP Address Mask Replies received for that local IP address MUST be (silently) ignored.

A host SHOULD make some reasonableness check on any address

mask it installs; see IMPLEMENTATION section below.

A system MUST NOT send an Address Mask Reply unless it is an authoritative agent for address masks. An authoritative agent may be a host or a gateway, but it MUST be explicitly configured as a address mask agent. Receiving an address mask via an Address Mask Reply does not give the receiver authority and MUST NOT be used as the basis for issuing Address Mask Replies.

With a statically configured address mask, there SHOULD be an additional configuration flag that determines whether the host is to act as an authoritative agent for this mask, i.e., whether it will answer Address Mask Request messages using this mask.

If it is configured as an agent, the host MUST broadcast an Address Mask Reply for the mask on the appropriate interface when it initializes.

See "System Initialization" in [INTRO:1] for more information about the use of Address Mask Request/Reply messages.

DISCUSSION

Hosts that casually send Address Mask Replies with invalid address masks have often been a serious nuisance. To prevent this, Address Mask Replies ought to be sent only by authoritative agents that have been selected by explicit administrative action.

When an authoritative agent receives an Address Mask Request message, it will send a unicast Address Mask Reply to the source IP address. If the network part of this address is zero (see (a) and (b) in 3.2.1.3), the Reply will be broadcast.

Getting no reply to its Address Mask Request messages, a host will assume there is no agent and use an unsubnetted mask, but the agent may be only temporarily unreachable. An agent will broadcast an unsolicited Address Mask Reply whenever it initializes, in order to update the masks of all hosts that have initialized in the meantime.

IMPLEMENTATION:

The following reasonableness check on an address mask is suggested: the mask is not all 1 bits, and it is

either zero or else the 8 highest-order bits are on.

3.2.3 Internet Group Management Protocol IGMP

IGMP [IP:4] is a protocol used between hosts and gateways on a single network to establish hosts' membership in particular multicast groups. The gateways use this information, in conjunction with a multicast routing protocol, to support IP multicasting across the Internet.

At this time, implementation of IGMP is OPTIONAL; see Section 3.3.7 for more information. Without IGMP, a host can still participate in multicasting local to its connected networks.

3.3 SPECIFIC ISSUES

3.3.1 Routing Outbound Datagrams

The IP layer chooses the correct next hop for each datagram it sends. If the destination is on a connected network, the datagram is sent directly to the destination host; otherwise, it has to be routed to a gateway on a connected network.

3.3.1.1 Local/Remote Decision

To decide if the destination is on a connected network, the following algorithm MUST be used [see IP:3]:

(a) The address mask (particular to a local IP address for a multihomed host) is a 32-bit mask that selects the network number and subnet number fields of the corresponding IP address.

(b) If the IP destination address bits extracted by the address mask match the IP source address bits extracted by the same mask, then the destination is on the corresponding connected network, and the datagram is to be transmitted directly to the destination host.

(c) If not, then the destination is accessible only through a gateway. Selection of a gateway is described below (3.3.1.2).

A special-case destination address is handled as follows:

* For a limited broadcast or a multicast address, simply pass the datagram to the link layer for the appropriate interface.

* For a (network or subnet) directed broadcast, the datagram can use the standard routing algorithms.

The host IP layer MUST operate correctly in a minimal network environment, and in particular, when there are no gateways. For example, if the IP layer of a host insists on finding at least one gateway to initialize, the host will be unable to operate on a single isolated broadcast net.

3.3.1.2 Gateway Selection

To efficiently route a series of datagrams to the same destination, the source host MUST keep a "route cache" of mappings to next-hop gateways. A host uses the following basic algorithm on this cache to route a datagram; this algorithm is designed to put the primary routing burden on the gateways [IP:11].

(a) If the route cache contains no information for a particular destination, the host chooses a "default" gateway and sends the datagram to it. It also builds a corresponding Route Cache entry.

(b) If that gateway is not the best next hop to the destination, the gateway will forward the datagram to the best next-hop gateway and return an ICMP Redirect message to the source host.

(c) When it receives a Redirect, the host updates the next-hop gateway in the appropriate route cache entry, so later datagrams to the same destination will go directly to the best gateway.

Since the subnet mask appropriate to the destination address is generally not known, a Network Redirect message SHOULD be treated identically to a Host Redirect message; i.e., the cache entry for the destination host (only) would be updated (or created, if an entry for that host did not exist) for the new gateway.

DISCUSSION:
This recommendation is to protect against gateways that erroneously send Network Redirects for a subnetted network, in violation of the gateway requirements [INTRO:2].

When there is no route cache entry for the destination host address (and the destination is not on the connected

network), the IP layer MUST pick a gateway from its list of "default" gateways. The IP layer MUST support multiple default gateways.

As an extra feature, a host IP layer MAY implement a table of "static routes". Each such static route MAY include a flag specifying whether it may be overridden by ICMP Redirects.

DISCUSSION:

A host generally needs to know at least one default gateway to get started. This information can be obtained from a configuration file or else from the host startup sequence, e.g., the BOOTP protocol (see [INTRO:1]).

It has been suggested that a host can augment its list of default gateways by recording any new gateways it learns about. For example, it can record every gateway to which it is ever redirected. Such a feature, while possibly useful in some circumstances, may cause problems in other cases (e.g., gateways are not all equal), and it is not recommended.

A static route is typically a particular preset mapping from destination host or network into a particular next-hop gateway; it might also depend on the Type-of-Service (see next section). Static routes would be set up by system administrators to override the normal automatic routing mechanism, to handle exceptional situations. However, any static routing information is a potential source of failure as configurations change or equipment fails.

3.3.1.3 Route Cache

Each route cache entry needs to include the following fields:

(1) Local IP address (for a multihomed host)

(2) Destination IP address

(3) Type(s)-of-Service

(4) Next-hop gateway IP address

Field (2) MAY be the full IP address of the destination

host, or only the destination network number. Field (3), the TOS, SHOULD be included.

See Section 3.3.4.2 for a discussion of the implications of multihoming for the lookup procedure in this cache.

DISCUSSION:

Including the Type-of-Service field in the route cache and considering it in the host route algorithm will provide the necessary mechanism for the future when Type-of-Service routing is commonly used in the Internet. See Section 3.2.1.6.

Each route cache entry defines the endpoints of an Internet path. Although the connecting path may change dynamically in an arbitrary way, the transmission characteristics of the path tend to remain approximately constant over a time period longer than a single typical host-host transport connection. Therefore, a route cache entry is a natural place to cache data on the properties of the path. Examples of such properties might be the maximum unfragmented datagram size (see Section 3.3.3), or the average round-trip delay measured by a transport protocol. This data will generally be both gathered and used by a higher layer protocol, e.g., by TCP, or by an application using UDP. Experiments are currently in progress on caching path properties in this manner.

There is no consensus on whether the route cache should be keyed on destination host addresses alone, or allow both host and network addresses. Those who favor the use of only host addresses argue that:

(1) As required in Section 3.3.1.2, Redirect messages will generally result in entries keyed on destination host addresses; the simplest and most general scheme would be to use host addresses always.

(2) The IP layer may not always know the address mask for a network address in a complex subnetted environment.

(3) The use of only host addresses allows the destination address to be used as a pure 32-bit number, which may allow the Internet architecture to be more easily extended in the future without

any change to the hosts.

The opposing view is that allowing a mixture of destination hosts and networks in the route cache:

(1) Saves memory space.

(2) Leads to a simpler data structure, easily combining the cache with the tables of default and static routes (see below).

(3) Provides a more useful place to cache path properties, as discussed earlier.

IMPLEMENTATION:

The cache needs to be large enough to include entries for the maximum number of destination hosts that may be in use at one time.

A route cache entry may also include control information used to choose an entry for replacement. This might take the form of a "recently used" bit, a use count, or a last-used timestamp, for example. It is recommended that it include the time of last modification of the entry, for diagnostic purposes.

An implementation may wish to reduce the overhead of scanning the route cache for every datagram to be transmitted. This may be accomplished with a hash table to speed the lookup, or by giving a connection-oriented transport protocol a "hint" or temporary handle on the appropriate cache entry, to be passed to the IP layer with each subsequent datagram.

Although we have described the route cache, the lists of default gateways, and a table of static routes as conceptually distinct, in practice they may be combined into a single "routing table" data structure.

3.3.1.4 Dead Gateway Detection

The IP layer MUST be able to detect the failure of a "next-hop" gateway that is listed in its route cache and to choose an alternate gateway (see Section 3.3.1.5).

Dead gateway detection is covered in some detail in RFC-816 [IP:11]. Experience to date has not produced a complete

algorithm which is totally satisfactory, though it has identified several forbidden paths and promising techniques.

* A particular gateway SHOULD NOT be used indefinitely in the absence of positive indications that it is functioning.

* Active probes such as "pinging" (i.e., using an ICMP Echo Request/Reply exchange) are expensive and scale poorly. In particular, hosts MUST NOT actively check the status of a first-hop gateway by simply pinging the gateway continuously.

* Even when it is the only effective way to verify a gateway's status, pinging MUST be used only when traffic is being sent to the gateway and when there is no other positive indication to suggest that the gateway is functioning.

* To avoid pinging, the layers above and/or below the Internet layer SHOULD be able to give "advice" on the status of route cache entries when either positive (gateway OK) or negative (gateway dead) information is available.

DISCUSSION:

If an implementation does not include an adequate mechanism for detecting a dead gateway and re-routing, a gateway failure may cause datagrams to apparently vanish into a "black hole". This failure can be extremely confusing for users and difficult for network personnel to debug.

The dead-gateway detection mechanism must not cause unacceptable load on the host, on connected networks, or on first-hop gateway(s). The exact constraints on the timeliness of dead gateway detection and on acceptable load may vary somewhat depending on the nature of the host's mission, but a host generally needs to detect a failed first-hop gateway quickly enough that transport-layer connections will not break before an alternate gateway can be selected.

Passing advice from other layers of the protocol stack complicates the interfaces between the layers, but it is the preferred approach to dead gateway detection. Advice can come from almost any part of the IP/TCP

architecture, but it is expected to come primarily from the transport and link layers. Here are some possible sources for gateway advice:

- TCP or any connection-oriented transport protocol should be able to give negative advice, e.g., triggered by excessive retransmissions.

- TCP may give positive advice when (new) data is acknowledged. Even though the route may be asymmetric, an ACK for new data proves that the acknowleged data must have been transmitted successfully.

- An ICMP Redirect message from a particular gateway should be used as positive advice about that gateway.

- Link-layer information that reliably detects and reports host failures (e.g., ARPANET Destination Dead messages) should be used as negative advice.

- Failure to ARP or to re-validate ARP mappings may be used as negative advice for the corresponding IP address.

- Packets arriving from a particular link-layer address are evidence that the system at this address is alive. However, turning this information into advice about gateways requires mapping the link-layer address into an IP address, and then checking that IP address against the gateways pointed to by the route cache. This is probably prohibitively inefficient.

Note that positive advice that is given for every datagram received may cause unacceptable overhead in the implementation.

While advice might be passed using required arguments in all interfaces to the IP layer, some transport and application layer protocols cannot deduce the correct advice. These interfaces must therefore allow a neutral value for advice, since either always-positive or always-negative advice leads to incorrect behavior.

There is another technique for dead gateway detection that has been commonly used but is not recommended.

This technique depends upon the host passively receiving ("wiretapping") the Interior Gateway Protocol (IGP) datagrams that the gateways are broadcasting to each other. This approach has the drawback that a host needs to recognize all the interior gateway protocols that gateways may use (see [INTRO:2]). In addition, it only works on a broadcast network.

At present, pinging (i.e., using ICMP Echo messages) is the mechanism for gateway probing when absolutely required. A successful ping guarantees that the addressed interface and its associated machine are up, but it does not guarantee that the machine is a gateway as opposed to a host. The normal inference is that if a Redirect or other evidence indicates that a machine was a gateway, successful pings will indicate that the machine is still up and hence still a gateway. However, since a host silently discards packets that a gateway would forward or redirect, this assumption could sometimes fail. To avoid this problem, a new ICMP message under development will ask "are you a gateway?"

IMPLEMENTATION:

The following specific algorithm has been suggested:

- Associate a "reroute timer" with each gateway pointed to by the route cache. Initialize the timer to a value Tr, which must be small enough to allow detection of a dead gateway before transport connections time out.

- Positive advice would reset the reroute timer to Tr. Negative advice would reduce or zero the reroute timer.

- Whenever the IP layer used a particular gateway to route a datagram, it would check the corresponding reroute timer. If the timer had expired (reached zero), the IP layer would send a ping to the gateway, followed immediately by the datagram.

- The ping (ICMP Echo) would be sent again if necessary, up to N times. If no ping reply was received in N tries, the gateway would be assumed to have failed, and a new first-hop gateway would be chosen for all cache entries pointing to the failed gateway.

Note that the size of Tr is inversely related to the amount of advice available. Tr should be large enough to insure that:

* Any pinging will be at a low level (e.g., <10%) of all packets sent to a gateway from the host, AND

* pinging is infrequent (e.g., every 3 minutes)

Since the recommended algorithm is concerned with the gateways pointed to by route cache entries, rather than the cache entries themselves, a two level data structure (perhaps coordinated with ARP or similar caches) may be desirable for implementing a route cache.

3.3.1.5 New Gateway Selection

If the failed gateway is not the current default, the IP layer can immediately switch to a default gateway. If it is the current default that failed, the IP layer MUST select a different default gateway (assuming more than one default is known) for the failed route and for establishing new routes.

DISCUSSION:

When a gateway does fail, the other gateways on the connected network will learn of the failure through some inter-gateway routing protocol. However, this will not happen instantaneously, since gateway routing protocols typically have a settling time of 30-60 seconds. If the host switches to an alternative gateway before the gateways have agreed on the failure, the new target gateway will probably forward the datagram to the failed gateway and send a Redirect back to the host pointing to the failed gateway (!). The result is likely to be a rapid oscillation in the contents of the host's route cache during the gateway settling period. It has been proposed that the dead-gateway logic should include some hysteresis mechanism to prevent such oscillations. However, experience has not shown any harm from such oscillations, since service cannot be restored to the host until the gateways' routing information does settle down.

IMPLEMENTATION:

One implementation technique for choosing a new default gateway is to simply round-robin among the default gateways in the host's list. Another is to rank the

gateways in priority order, and when the current default gateway is not the highest priority one, to "ping" the higher-priority gateways slowly to detect when they return to service. This pinging can be at a very low rate, e.g., 0.005 per second.

3.3.1.6 Initialization

The following information MUST be configurable:

(1) IP address(es).

(2) Address mask(s).

(3) A list of default gateways, with a preference level.

A manual method of entering this configuration data MUST be provided. In addition, a variety of methods can be used to determine this information dynamically; see the section on "Host Initialization" in [INTRO:1].

DISCUSSION:
Some host implementations use "wiretapping" of gateway protocols on a broadcast network to learn what gateways exist. A standard method for default gateway discovery is under development.

3.3.2 Reassembly

The IP layer MUST implement reassembly of IP datagrams.

We designate the largest datagram size that can be reassembled by EMTU_R ("Effective MTU to receive"); this is sometimes called the "reassembly buffer size". EMTU_R MUST be greater than or equal to 576, SHOULD be either configurable or indefinite, and SHOULD be greater than or equal to the MTU of the connected network(s).

DISCUSSION:
A fixed EMTU_R limit should not be built into the code because some application layer protocols require EMTU_R values larger than 576.

IMPLEMENTATION:
An implementation may use a contiguous reassembly buffer for each datagram, or it may use a more complex data structure that places no definite limit on the reassembled datagram size; in the latter case, EMTU_R is said to be

"indefinite".

Logically, reassembly is performed by simply copying each fragment into the packet buffer at the proper offset. Note that fragments may overlap if successive retransmissions use different packetizing but the same reassembly Id.

The tricky part of reassembly is the bookkeeping to determine when all bytes of the datagram have been reassembled. We recommend Clark's algorithm [IP:10] that requires no additional data space for the bookkeeping. However, note that, contrary to [IP:10], the first fragment header needs to be saved for inclusion in a possible ICMP Time Exceeded (Reassembly Timeout) message.

There MUST be a mechanism by which the transport layer can learn MMS_R, the maximum message size that can be received and reassembled in an IP datagram (see GET_MAXSIZES calls in Section 3.4). If EMTU_R is not indefinite, then the value of MMS_R is given by:

MMS_R = EMTU_R - 20

since 20 is the minimum size of an IP header.

There MUST be a reassembly timeout. The reassembly timeout value SHOULD be a fixed value, not set from the remaining TTL. It is recommended that the value lie between 60 seconds and 120 seconds. If this timeout expires, the partially-reassembled datagram MUST be discarded and an ICMP Time Exceeded message sent to the source host (if fragment zero has been received).

DISCUSSION:

The IP specification says that the reassembly timeout should be the remaining TTL from the IP header, but this does not work well because gateways generally treat TTL as a simple hop count rather than an elapsed time. If the reassembly timeout is too small, datagrams will be discarded unnecessarily, and communication may fail. The timeout needs to be at least as large as the typical maximum delay across the Internet. A realistic minimum reassembly timeout would be 60 seconds.

It has been suggested that a cache might be kept of round-trip times measured by transport protocols for various destinations, and that these values might be used to dynamically determine a reasonable reassembly timeout

value. Further investigation of this approach is required.

If the reassembly timeout is set too high, buffer resources in the receiving host will be tied up too long, and the MSL (Maximum Segment Lifetime) [TCP:1] will be larger than necessary. The MSL controls the maximum rate at which fragmented datagrams can be sent using distinct values of the 16-bit Ident field; a larger MSL lowers the maximum rate. The TCP specification [TCP:1] arbitrarily assumes a value of 2 minutes for MSL. This sets an upper limit on a reasonable reassembly timeout value.

3.3.3 Fragmentation

Optionally, the IP layer MAY implement a mechanism to fragment outgoing datagrams intentionally.

We designate by EMTU_S ("Effective MTU for sending") the maximum IP datagram size that may be sent, for a particular combination of IP source and destination addresses and perhaps TOS.

A host MUST implement a mechanism to allow the transport layer to learn MMS_S, the maximum transport-layer message size that may be sent for a given {source, destination, TOS} triplet (see GET_MAXSIZES call in Section 3.4). If no local fragmentation is performed, the value of MMS_S will be:

```
MMS_S = EMTU_S - <IP header size>
```

and EMTU_S must be less than or equal to the MTU of the network interface corresponding to the source address of the datagram. Note that <IP header size> in this equation will be 20, unless the IP reserves space to insert IP options for its own purposes in addition to any options inserted by the transport layer.

A host that does not implement local fragmentation MUST ensure that the transport layer (for TCP) or the application layer (for UDP) obtains MMS_S from the IP layer and does not send a datagram exceeding MMS_S in size.

It is generally desirable to avoid local fragmentation and to choose EMTU_S low enough to avoid fragmentation in any gateway along the path. In the absence of actual knowledge of the minimum MTU along the path, the IP layer SHOULD use EMTU_S <= 576 whenever the destination address is not on a connected network, and otherwise use the connected network's

MTU.

The MTU of each physical interface MUST be configurable.

A host IP layer implementation MAY have a configuration flag "All-Subnets-MTU", indicating that the MTU of the connected network is to be used for destinations on different subnets within the same network, but not for other networks. Thus, this flag causes the network class mask, rather than the subnet address mask, to be used to choose an EMTU_S. For a multihomed host, an "All-Subnets-MTU" flag is needed for each network interface.

DISCUSSION:

Picking the correct datagram size to use when sending data is a complex topic [IP:9].

(a) In general, no host is required to accept an IP datagram larger than 576 bytes (including header and data), so a host must not send a larger datagram without explicit knowledge or prior arrangement with the destination host. Thus, MMS_S is only an upper bound on the datagram size that a transport protocol may send; even when MMS_S exceeds 556, the transport layer must limit its messages to 556 bytes in the absence of other knowledge about the destination host.

(b) Some transport protocols (e.g., TCP) provide a way to explicitly inform the sender about the largest datagram the other end can receive and reassemble [IP:7]. There is no corresponding mechanism in the IP layer.

A transport protocol that assumes an EMTU_R larger than 576 (see Section 3.3.2), can send a datagram of this larger size to another host that implements the same protocol.

(c) Hosts should ideally limit their EMTU_S for a given destination to the minimum MTU of all the networks along the path, to avoid any fragmentation. IP fragmentation, while formally correct, can create a serious transport protocol performance problem, because loss of a single fragment means all the fragments in the segment must be retransmitted [IP:9].

Since nearly all networks in the Internet currently support an MTU of 576 or greater, we strongly recommend the use of 576 for datagrams sent to non-local networks.

It has been suggested that a host could determine the MTU over a given path by sending a zero-offset datagram fragment and waiting for the receiver to time out the reassembly (which cannot complete!) and return an ICMP Time Exceeded message. This message would include the largest remaining fragment header in its body. More direct mechanisms are being experimented with, but have not yet been adopted (see e.g., RFC-1063).

3.3.4 Local Multihoming

3.3.4.1 Introduction

A multihomed host has multiple IP addresses, which we may think of as "logical interfaces". These logical interfaces may be associated with one or more physical interfaces, and these physical interfaces may be connected to the same or different networks.

Here are some important cases of multihoming:

(a) Multiple Logical Networks

The Internet architects envisioned that each physical network would have a single unique IP network (or subnet) number. However, LAN administrators have sometimes found it useful to violate this assumption, operating a LAN with multiple logical networks per physical connected network.

If a host connected to such a physical network is configured to handle traffic for each of N different logical networks, then the host will have N logical interfaces. These could share a single physical interface, or might use N physical interfaces to the same network.

(b) Multiple Logical Hosts

When a host has multiple IP addresses that all have the same <Network-number> part (and the same <Subnet-number> part, if any), the logical interfaces are known as "logical hosts". These logical interfaces might share a single physical interface or might use separate

physical interfaces to the same physical network.

(c) Simple Multihoming

In this case, each logical interface is mapped into a separate physical interface and each physical interface is connected to a different physical network. The term "multihoming" was originally applied only to this case, but it is now applied more generally.

A host with embedded gateway functionality will typically fall into the simple multihoming case. Note, however, that a host may be simply multihomed without containing an embedded gateway, i.e., without forwarding datagrams from one connected network to another.

This case presents the most difficult routing problems. The choice of interface (i.e., the choice of first-hop network) may significantly affect performance or even reachability of remote parts of the Internet.

Finally, we note another possibility that is NOT multihoming: one logical interface may be bound to multiple physical interfaces, in order to increase the reliability or throughput between directly connected machines by providing alternative physical paths between them. For instance, two systems might be connected by multiple point-to-point links. We call this "link-layer multiplexing". With link-layer multiplexing, the protocols above the link layer are unaware that multiple physical interfaces are present; the link-layer device driver is responsible for multiplexing and routing packets across the physical interfaces.

In the Internet protocol architecture, a transport protocol instance ("entity") has no address of its own, but instead uses a single Internet Protocol (IP) address. This has implications for the IP, transport, and application layers, and for the interfaces between them. In particular, the application software may have to be aware of the multiple IP addresses of a multihomed host; in other cases, the choice can be made within the network software.

3.3.4.2 Multihoming Requirements

The following general rules apply to the selection of an IP source address for sending a datagram from a multihomed

host.

(1) If the datagram is sent in response to a received datagram, the source address for the response SHOULD be the specific-destination address of the request. See Sections 4.1.3.5 and 4.2.3.7 and the "General Issues" section of [INTRO:1] for more specific requirements on higher layers.

    Otherwise, a source address must be selected.

(2) An application MUST be able to explicitly specify the source address for initiating a connection or a request.

(3) In the absence of such a specification, the networking software MUST choose a source address. Rules for this choice are described below.

There are two key requirement issues related to multihoming:

(A) A host MAY silently discard an incoming datagram whose destination address does not correspond to the physical interface through which it is received.

(B) A host MAY restrict itself to sending (non-source-routed) IP datagrams only through the physical interface that corresponds to the IP source address of the datagrams.

DISCUSSION:

Internet host implementors have used two different conceptual models for multihoming, briefly summarized in the following discussion. This document takes no stand on which model is preferred; each seems to have a place. This ambivalence is reflected in the issues (A) and (B) being optional.

- Strong ES Model

  The Strong ES (End System, i.e., host) model emphasizes the host/gateway (ES/IS) distinction, and would therefore substitute MUST for MAY in issues (A) and (B) above. It tends to model a multihomed host as a set of logical hosts within the same physical host.

With respect to (A), proponents of the Strong ES model note that automatic Internet routing mechanisms could not route a datagram to a physical interface that did not correspond to the destination address.

Under the Strong ES model, the route computation for an outgoing datagram is the mapping:

```
route(src IP addr, dest IP addr, TOS)
                                    -> gateway
```

Here the source address is included as a parameter in order to select a gateway that is directly reachable on the corresponding physical interface. Note that this model logically requires that in general there be at least one default gateway, and preferably multiple defaults, for each IP source address.

o Weak ES Model

This view de-emphasizes the ES/IS distinction, and would therefore substitute MUST NOT for MAY in issues (A) and (B). This model may be the more natural one for hosts that wiretap gateway routing protocols, and is necessary for hosts that have embedded gateway functionality.

The Weak ES Model may cause the Redirect mechanism to fail. If a datagram is sent out a physical interface that does not correspond to the destination address, the first-hop gateway will not realize when it needs to send a Redirect. On the other hand, if the host has embedded gateway functionality, then it has routing information without listening to Redirects.

In the Weak ES model, the route computation for an outgoing datagram is the mapping:

```
route(dest IP addr, TOS) -> gateway, interface
```

3.3.4.3 Choosing a Source Address

DISCUSSION:

When it sends an initial connection request (e.g., a TCP "SYN" segment) or a datagram service request (e.g., a UDP-based query), the transport layer on a multihomed host needs to know which source address to use. If the application does not specify it, the transport layer must ask the IP layer to perform the conceptual mapping:

```
GET_SRCADDR(remote IP addr, TOS)
                              -> local IP address
```

Here TOS is the Type-of-Service value (see Section 3.2.1.6), and the result is the desired source address. The following rules are suggested for implementing this mapping:

(a) If the remote Internet address lies on one of the (sub-) nets to which the host is directly connected, a corresponding source address may be chosen, unless the corresponding interface is known to be down.

(b) The route cache may be consulted, to see if there is an active route to the specified destination network through any network interface; if so, a local IP address corresponding to that interface may be chosen.

(c) The table of static routes, if any (see Section 3.3.1.2) may be similarly consulted.

(d) The default gateways may be consulted. If these gateways are assigned to different interfaces, the interface corresponding to the gateway with the highest preference may be chosen.

In the future, there may be a defined way for a multihomed host to ask the gateways on all connected networks for advice about the best network to use for a given destination.

IMPLEMENTATION:

It will be noted that this process is essentially the same as datagram routing (see Section 3.3.1), and therefore hosts may be able to combine the

implementation of the two functions.

3.3.5 Source Route Forwarding

Subject to restrictions given below, a host MAY be able to act as an intermediate hop in a source route, forwarding a source-routed datagram to the next specified hop.

However, in performing this gateway-like function, the host MUST obey all the relevant rules for a gateway forwarding source-routed datagrams [INTRO:2]. This includes the following specific provisions, which override the corresponding host provisions given earlier in this document:

(A) TTL (ref. Section 3.2.1.7)

The TTL field MUST be decremented and the datagram perhaps discarded as specified for a gateway in [INTRO:2].

(B) ICMP Destination Unreachable (ref. Section 3.2.2.1)

A host MUST be able to generate Destination Unreachable messages with the following codes:

4 (Fragmentation Required but DF Set) when a source-routed datagram cannot be fragmented to fit into the target network;

5 (Source Route Failed) when a source-routed datagram cannot be forwarded, e.g., because of a routing problem or because the next hop of a strict source route is not on a connected network.

(C) IP Source Address (ref. Section 3.2.1.3)

A source-routed datagram being forwarded MAY (and normally will) have a source address that is not one of the IP addresses of the forwarding host.

(D) Record Route Option (ref. Section 3.2.1.8d)

A host that is forwarding a source-routed datagram containing a Record Route option MUST update that option, if it has room.

(E) Timestamp Option (ref. Section 3.2.1.8e)

A host that is forwarding a source-routed datagram

containing a Timestamp Option MUST add the current timestamp to that option, according to the rules for this option.

To define the rules restricting host forwarding of source-routed datagrams, we use the term "local source-routing" if the next hop will be through the same physical interface through which the datagram arrived; otherwise, it is "non-local source-routing".

- A host is permitted to perform local source-routing without restriction.
- A host that supports non-local source-routing MUST have a configurable switch to disable forwarding, and this switch MUST default to disabled.
- The host MUST satisfy all gateway requirements for configurable policy filters [INTRO:2] restricting non-local forwarding.

If a host receives a datagram with an incomplete source route but does not forward it for some reason, the host SHOULD return an ICMP Destination Unreachable (code 5, Source Route Failed) message, unless the datagram was itself an ICMP error message.

3.3.6 Broadcasts

Section 3.2.1.3 defined the four standard IP broadcast address forms:

Limited Broadcast: {-1, -1}

Directed Broadcast: {<Network-number>,-1}

Subnet Directed Broadcast:
{<Network-number>,<Subnet-number>,-1}

All-Subnets Directed Broadcast: {<Network-number>,-1,-1}

A host MUST recognize any of these forms in the destination address of an incoming datagram.

There is a class of hosts* that use non-standard broadcast address forms, substituting 0 for -1. All hosts SHOULD

---

*4.2BSD Unix and its derivatives, but not 4.3BSD.

recognize and accept any of these non-standard broadcast addresses as the destination address of an incoming datagram. A host MAY optionally have a configuration option to choose the 0 or the -1 form of broadcast address, for each physical interface, but this option SHOULD default to the standard (-1) form.

When a host sends a datagram to a link-layer broadcast address, the IP destination address MUST be a legal IP broadcast or IP multicast address.

A host SHOULD silently discard a datagram that is received via a link-layer broadcast (see Section 2.4) but does not specify an IP multicast or broadcast destination address.

Hosts SHOULD use the Limited Broadcast address to broadcast to a connected network.

DISCUSSION:

Using the Limited Broadcast address instead of a Directed Broadcast address may improve system robustness. Problems are often caused by machines that do not understand the plethora of broadcast addresses (see Section 3.2.1.3), or that may have different ideas about which broadcast addresses are in use. The prime example of the latter is machines that do not understand subnetting but are attached to a subnetted net. Sending a Subnet Broadcast for the connected network will confuse those machines, which will see it as a message to some other host.

There has been discussion on whether a datagram addressed to the Limited Broadcast address ought to be sent from all the interfaces of a multihomed host. This specification takes no stand on the issue.

3.3.7 IP Multicasting

A host SHOULD support local IP multicasting on all connected networks for which a mapping from Class D IP addresses to link-layer addresses has been specified (see below). Support for local IP multicasting includes sending multicast datagrams, joining multicast groups and receiving multicast datagrams, and leaving multicast groups. This implies support for all of [IP:4] except the IGMP protocol itself, which is OPTIONAL.

DISCUSSION:

IGMP provides gateways that are capable of multicast routing with the information required to support IP multicasting across multiple networks. At this time, multicast-routing gateways are in the experimental stage and are not widely available. For hosts that are not connected to networks with multicast-routing gateways or that do not need to receive multicast datagrams originating on other networks, IGMP serves no purpose and is therefore optional for now. However, the rest of [IP:4] is currently recommended for the purpose of providing IP-layer access to local network multicast addressing, as a preferable alternative to local broadcast addressing. It is expected that IGMP will become recommended at some future date, when multicast-routing gateways have become more widely available.

If IGMP is not implemented, a host SHOULD still join the "all-hosts" group (224.0.0.1) when the IP layer is initialized and remain a member for as long as the IP layer is active.

DISCUSSION:

Joining the "all-hosts" group will support strictly local uses of multicasting, e.g., a gateway discovery protocol, even if IGMP is not implemented.

The mapping of IP Class D addresses to local addresses is currently specified for the following types of networks:

- Ethernet/IEEE 802.3, as defined in [IP:4].
- Any network that supports broadcast but not multicast, addressing: all IP Class D addresses map to the local broadcast address.
- Any type of point-to-point link (e.g., SLIP or HDLC links): no mapping required. All IP multicast datagrams are sent as-is, inside the local framing.

Mappings for other types of networks will be specified in the future.

A host SHOULD provide a way for higher-layer protocols or applications to determine which of the host’s connected network(s) support IP multicast addressing.

3.3.8 Error Reporting

Wherever practical, hosts MUST return ICMP error datagrams on detection of an error, except in those cases where returning an ICMP error message is specifically prohibited.

DISCUSSION:
A common phenomenon in datagram networks is the "black hole disease": datagrams are sent out, but nothing comes back. Without any error datagrams, it is difficult for the user to figure out what the problem is.

3.4 INTERNET/TRANSPORT LAYER INTERFACE

The interface between the IP layer and the transport layer MUST provide full access to all the mechanisms of the IP layer, including options, Type-of-Service, and Time-to-Live. The transport layer MUST either have mechanisms to set these interface parameters, or provide a path to pass them through from an application, or both.

DISCUSSION:
Applications are urged to make use of these mechanisms where applicable, even when the mechanisms are not currently effective in the Internet (e.g., TOS). This will allow these mechanisms to be immediately useful when they do become effective, without a large amount of retrofitting of host software.

We now describe a conceptual interface between the transport layer and the IP layer, as a set of procedure calls. This is an extension of the information in Section 3.3 of RFC-791 [IP:1].

* Send Datagram

```
SEND(src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt
     => result )
```

where the parameters are defined in RFC-791. Passing an Id parameter is optional; see Section 3.2.1.5.

* Receive Datagram

```
RECV(BufPTR, prot
     => result, src, dst, SpecDest, TOS, len, opt)
```

All the parameters are defined in RFC-791, except for:

SpecDest = specific-destination address of datagram
(defined in Section 3.2.1.3)

The result parameter dst contains the datagram's destination address. Since this may be a broadcast or multicast address, the SpecDest parameter (not shown in RFC-791) MUST be passed. The parameter opt contains all the IP options received in the datagram; these MUST also be passed to the transport layer.

* Select Source Address

GET_SRCADDR(remote, TOS) -> local

remote = remote IP address
TOS = Type-of-Service
local = local IP address

See Section 3.3.4.3.

* Find Maximum Datagram Sizes

GET_MAXSIZES(local, remote, TOS) -> MMS_R, MMS_S

MMS_R = maximum receive transport-message size.
MMS_S = maximum send transport-message size.
(local, remote, TOS defined above)

See Sections 3.3.2 and 3.3.3.

* Advice on Delivery Success

ADVISE_DELIVPROB(sense, local, remote, TOS)

Here the parameter sense is a 1-bit flag indicating whether positive or negative advice is being given; see the discussion in Section 3.3.1.4. The other parameters were defined earlier.

* Send ICMP Message

SEND_ICMP(src, dst, TOS, TTL, BufPTR, len, Id, DF, opt)
-> result

(Parameters defined in RFC-791).

Passing an Id parameter is optional; see Section 3.2.1.5. The transport layer MUST be able to send certain ICMP messages: Port Unreachable or any of the query-type messages. This function could be considered to be a special case of the SEND() call, of course; we describe it separately for clarity.

* Receive ICMP Message

RECV_ICMP(BufPTR) -> result, src, dst, len, opt

(Parameters defined in RFC-791).

The IP layer MUST pass certain ICMP messages up to the appropriate transport-layer routine. This function could be considered to be a special case of the RECV() call, of course; we describe it separately for clarity.

For an ICMP error message, the data that is passed up MUST include the original Internet header plus all the octets of the original message that are included in the ICMP message. This data will be used by the transport layer to locate the connection state information, if any.

In particular, the following ICMP messages are to be passed up:

- Destination Unreachable
- Source Quench
- Echo Reply (to ICMP user interface, unless the Echo Request originated in the IP layer)
- Timestamp Reply (to ICMP user interface)
- Time Exceeded

DISCUSSION:
In the future, there may be additions to this interface to pass path data (see Section 3.3.1.3) between the IP and transport layers.

3.5 INTERNET LAYER REQUIREMENTS SUMMARY

FEATURE	SECTION	SHOULD	SHOULD NOT	MUST	MUST NOT	OPTIONAL	FOOTNOTE
Implement IP and ICMP	3.1	x					
Handle remote multihoming in application layer	3.1	x					
Support local multihoming	3.1			x			
Meet gateway specs if forward datagrams	3.1	x					
Configuration switch for embedded gateway	3.1	x					1
Config switch default to non-gateway	3.1	x					1
Auto-config based on number of interfaces	3.1				x		1
Able to log discarded datagrams	3.1		x				
Record in counter	3.1		x				
Silently discard Version != 4	3.2.1.1	x					
Verify IP checksum, silently discard bad dgram	3.2.1.2	x					
Addressing:							
Subnet addressing (RFC-950)	3.2.1.3	x					
Src address must be host's own IP address	3.2.1.3	x					
Silently discard datagram with bad dest addr	3.2.1.3	x					
Silently discard datagram with bad src addr	3.2.1.3	x					
Support reassembly	3.2.1.4	x					
Retain same Id field in identical datagram	3.2.1.5			x			
TOS:							
Allow transport layer to set TOS	3.2.1.6	x					
Pass received TOS up to transport layer	3.2.1.6		x				
Use RFC-795 link-layer mappings for TOS	3.2.1.6				x		
TTL:							
Send packet with TTL of 0	3.2.1.7					x	
Discard received packets with TTL < 2	3.2.1.7					x	
Allow transport layer to set TTL	3.2.1.7	x					
Fixed TTL is configurable	3.2.1.7	x					
IP Options:							
Allow transport layer to send IP options	3.2.1.8	x					
Pass all IP options rcvd to higher layer	3.2.1.8	x					

IP layer silently ignore unknown options	3.2.1.8	x				
Security option	3.2.1.8a			x		
Send Stream Identifier option	3.2.1.8b				x	
Silently ignore Stream Identifer option	3.2.1.8b	x				
Record Route option	3.2.1.8d			x		
Timestamp option	3.2.1.8e			x		
Source Route Option:						
Originate & terminate Source Route options	3.2.1.8c	x				
Datagram with completed SR passed up to TL	3.2.1.8c	x				
Build correct (non-redundant) return route	3.2.1.8c	x				
Send multiple SR options in one header	3.2.1.8c					x
ICMP:						
Silently discard ICMP msg with unknown type	3.2.2	x				
Include more than 8 octets of orig datagram	3.2.2			x		
Included octets same as received	3.2.2	x				
Demux ICMP Error to transport protocol	3.2.2	x				
Send ICMP error message with TOS=0	3.2.2		x			
Send ICMP error message for:						
- ICMP error msg	3.2.2					x
- IP b'cast or IP m'cast	3.2.2					x
- Link-layer b'cast	3.2.2					x
- Non-initial fragment	3.2.2					x
- Datagram with non-unique src address	3.2.2					x
Return ICMP error msgs (when not prohibited)	3.3.8	x				
Dest Unreachable:						
Generate Dest Unreachable (code 2/3)	3.2.2.1		x			
Pass ICMP Dest Unreachable to higher layer	3.2.2.1	x				
Higher layer act on Dest Unreach	3.2.2.1		x			
Interpret Dest Unreach as only hint	3.2.2.1	x				
Redirect:						
Host send Redirect	3.2.2.2				x	
Update route cache when recv Redirect	3.2.2.2	x				
Handle both Host and Net Redirects	3.2.2.2	x				
Discard illegal Redirect	3.2.2.2		x			
Source Quench:						
Send Source Quench if buffering exceeded	3.2.2.3			x		
Pass Source Quench to higher layer	3.2.2.3	x				
Higher layer act on Source Quench	3.2.2.3		x			
Time Exceeded: pass to higher layer	3.2.2.4	x				
Parameter Problem:						
Send Parameter Problem messages	3.2.2.5		x			
Pass Parameter Problem to higher layer	3.2.2.5	x				
Report Parameter Problem to user	3.2.2.5			x		
ICMP Echo Request or Reply:						
Echo server and Echo client	3.2.2.6	x				

Echo client	3.2.2.6		x				
Discard Echo Request to broadcast address	3.2.2.6			x			
Discard Echo Request to multicast address	3.2.2.6			x			
Use specific-dest addr as Echo Reply src	3.2.2.6	x					
Send same data in Echo Reply	3.2.2.6	x					
Pass Echo Reply to higher layer	3.2.2.6	x					
Reflect Record Route, Time Stamp options	3.2.2.6		x				
Reverse and reflect Source Route option	3.2.2.6	x					
ICMP Information Request or Reply:	3.2.2.7				x		
ICMP Timestamp and Timestamp Reply:	3.2.2.8			x			
Minimize delay variability	3.2.2.8		x				1
Silently discard b'cast Timestamp	3.2.2.8			x			1
Silently discard m'cast Timestamp	3.2.2.8			x			1
Use specific-dest addr as TS Reply src	3.2.2.8	x					1
Reflect Record Route, Time Stamp options	3.2.2.6		x				1
Reverse and reflect Source Route option	3.2.2.8	x					1
Pass Timestamp Reply to higher layer	3.2.2.8	x					1
Obey rules for "standard value"	3.2.2.8	x					1
ICMP Address Mask Request and Reply:							
Addr Mask source configurable	3.2.2.9	x					
Support static configuration of addr mask	3.2.2.9	x					
Get addr mask dynamically during booting	3.2.2.9			x			
Get addr via ICMP Addr Mask Request/Reply	3.2.2.9			x			
Retransmit Addr Mask Req if no Reply	3.2.2.9	x					3
Assume default mask if no Reply	3.2.2.9		x				3
Update address mask from first Reply only	3.2.2.9	x					3
Reasonableness check on Addr Mask	3.2.2.9		x				
Send unauthorized Addr Mask Reply msgs	3.2.2.9					x	
Explicitly configured to be agent	3.2.2.9	x					
Static config=> Addr-Mask-Authoritative flag	3.2.2.9		x				
Broadcast Addr Mask Reply when init.	3.2.2.9	x					3
ROUTING OUTBOUND DATAGRAMS:							
Use address mask in local/remote decision	3.3.1.1	x					
Operate with no gateways on conn network	3.3.1.1	x					
Maintain "route cache" of next-hop gateways	3.3.1.2	x					
Treat Host and Net Redirect the same	3.3.1.2		x				
If no cache entry, use default gateway	3.3.1.2	x					
Support multiple default gateways	3.3.1.2	x					
Provide table of static routes	3.3.1.2			x			
Flag: route overridable by Redirects	3.3.1.2			x			
Key route cache on host, not net address	3.3.1.3			x			
Include TOS in route cache	3.3.1.3		x				
Able to detect failure of next-hop gateway	3.3.1.4	x					
Assume route is good forever	3.3.1.4				x		

Ping gateways continuously	3.3.1.4					x	
Ping only when traffic being sent	3.3.1.4	x					
Ping only when no positive indication	3.3.1.4	x					
Higher and lower layers give advice	3.3.1.4		x				
Switch from failed default g'way to another	3.3.1.5	x					
Manual method of entering config info	3.3.1.6	x					
REASSEMBLY and FRAGMENTATION:							
Able to reassemble incoming datagrams	3.3.2	x					
At least 576 byte datagrams	3.3.2	x					
EMTU_R configurable or indefinite	3.3.2		x				
Transport layer able to learn MMS_R	3.3.2	x					
Send ICMP Time Exceeded on reassembly timeout	3.3.2	x					
Fixed reassembly timeout value	3.3.2		x				
Pass MMS_S to higher layers	3.3.3	x					
Local fragmentation of outgoing packets	3.3.3			x			
Else don't send bigger than MMS_S	3.3.3	x					
Send max 576 to off-net destination	3.3.3		x				
All-Subnets-MTU configuration flag	3.3.3			x			
MULTIHOMING:							
Reply with same addr as spec-dest addr	3.3.4.2		x				
Allow application to choose local IP addr	3.3.4.2	x					
Silently discard d'gram in "wrong" interface	3.3.4.2			x			
Only send d'gram through "right" interface	3.3.4.2			x			4
SOURCE-ROUTE FORWARDING:							
Forward datagram with Source Route option	3.3.5			x			1
Obey corresponding gateway rules	3.3.5	x					1
Update TTL by gateway rules	3.3.5	x					1
Able to generate ICMP err code 4, 5	3.3.5	x					1
IP src addr not local host	3.3.5			x			1
Update Timestamp, Record Route options	3.3.5	x					1
Configurable switch for non-local SRing	3.3.5	x					1
Defaults to OFF	3.3.5	x					1
Satisfy gwy access rules for non-local SRing	3.3.5	x					1
If not forward, send Dest Unreach (cd 5)	3.3.5		x				2
BROADCAST:							
Broadcast addr as IP source addr	3.2.1.3					x	
Receive 0 or -1 broadcast formats OK	3.3.6		x				
Config'ble option to send 0 or -1 b'cast	3.3.6			x			
Default to -1 broadcast	3.3.6		x				
Recognize all broadcast address formats	3.3.6	x					
Use IP b'cast/m'cast addr in link-layer b'cast	3.3.6	x					
Silently discard link-layer-only b'cast dg's	3.3.6		x				
Use Limited Broadcast addr for connected net	3.3.6		x				

MULTICAST:						
Support local IP multicasting (RFC-1112)	3.3.7		x			
Support IGMP (RFC-1112)	3.3.7			x		
Join all-hosts group at startup	3.3.7		x			
Higher layers learn i'face m'cast capability	3.3.7		x			
INTERFACE:						
Allow transport layer to use all IP mechanisms	3.4	x				
Pass interface ident up to transport layer	3.4	x				
Pass all IP options up to transport layer	3.4	x				
Transport layer can send certain ICMP messages	3.4	x				
Pass spec'd ICMP messages up to transp. layer	3.4	x				
Include IP hdr+8 octets or more from orig.	3.4	x				
Able to leap tall buildings at a single bound	3.5		x			

Footnotes:

(1) Only if feature is implemented.

(2) This requirement is overruled if datagram is an ICMP error message.

(3) Only if feature is implemented and is configured "on".

(4) Unless has embedded gateway functionality or is source routed.

4. TRANSPORT PROTOCOLS

4.1 USER DATAGRAM PROTOCOL -- UDP

4.1.1 INTRODUCTION

The User Datagram Protocol UDP [UDP:1] offers only a minimal transport service -- non-guaranteed datagram delivery -- and gives applications direct access to the datagram service of the IP layer. UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP.

UDP is almost a null protocol; the only services it provides over IP are checksumming of data and multiplexing by port number. Therefore, an application program running over UDP must deal directly with end-to-end communication problems that a connection-oriented protocol would have handled -- e.g., retransmission for reliable delivery, packetization and reassembly, flow control, congestion avoidance, etc., when these are required. The fairly complex coupling between IP and TCP will be mirrored in the coupling between UDP and many applications using UDP.

4.1.2 PROTOCOL WALK-THROUGH

There are no known errors in the specification of UDP.

4.1.3 SPECIFIC ISSUES

4.1.3.1 Ports

UDP well-known ports follow the same rules as TCP well-known ports; see Section 4.2.2.1 below.

If a datagram arrives addressed to a UDP port for which there is no pending LISTEN call, UDP SHOULD send an ICMP Port Unreachable message.

4.1.3.2 IP Options

UDP MUST pass any IP option that it receives from the IP layer transparently to the application layer.

An application MUST be able to specify IP options to be sent in its UDP datagrams, and UDP MUST pass these options to the IP layer.

DISCUSSION:

At present, the only options that need be passed through UDP are Source Route, Record Route, and Time Stamp. However, new options may be defined in the future, and UDP need not and should not make any assumptions about the format or content of options it passes to or from the application; an exception to this might be an IP-layer security option.

An application based on UDP will need to obtain a source route from a request datagram and supply a reversed route for sending the corresponding reply.

4.1.3.3 ICMP Messages

UDP MUST pass to the application layer all ICMP error messages that it receives from the IP layer. Conceptually at least, this may be accomplished with an upcall to the ERROR_REPORT routine (see Section 4.2.4.1).

DISCUSSION:

Note that ICMP error messages resulting from sending a UDP datagram are received asynchronously. A UDP-based application that wants to receive ICMP error messages is responsible for maintaining the state necessary to demultiplex these messages when they arrive; for example, the application may keep a pending receive operation for this purpose. The application is also responsible to avoid confusion from a delayed ICMP error message resulting from an earlier use of the same port(s).

4.1.3.4 UDP Checksums

A host MUST implement the facility to generate and validate UDP checksums. An application MAY optionally be able to control whether a UDP checksum will be generated, but it MUST default to checksumming on.

If a UDP datagram is received with a checksum that is non-zero and invalid, UDP MUST silently discard the datagram. An application MAY optionally be able to control whether UDP datagrams without checksums should be discarded or passed to the application.

DISCUSSION:

Some applications that normally run only across local area networks have chosen to turn off UDP checksums for

efficiency. As a result, numerous cases of undetected errors have been reported. The advisability of ever turning off UDP checksumming is very controversial.

IMPLEMENTATION:
There is a common implementation error in UDP checksums. Unlike the TCP checksum, the UDP checksum is optional; the value zero is transmitted in the checksum field of a UDP header to indicate the absence of a checksum. If the transmitter really calculates a UDP checksum of zero, it must transmit the checksum as all 1's (65535). No special action is required at the receiver, since zero and 65535 are equivalent in 1's complement arithmetic.

4.1.3.5 UDP Multihoming

When a UDP datagram is received, its specific-destination address MUST be passed up to the application layer.

An application program MUST be able to specify the IP source address to be used for sending a UDP datagram or to leave it unspecified (in which case the networking software will choose an appropriate source address). There SHOULD be a way to communicate the chosen source address up to the application layer (e.g, so that the application can later receive a reply datagram only from the corresponding interface).

DISCUSSION:
A request/response application that uses UDP should use a source address for the response that is the same as the specific destination address of the request. See the "General Issues" section of [INTRO:1].

4.1.3.6 Invalid Addresses

A UDP datagram received with an invalid IP source address (e.g., a broadcast or multicast address) must be discarded by UDP or by the IP layer (see Section 3.2.1.3).

When a host sends a UDP datagram, the source address MUST be (one of) the IP address(es) of the host.

4.1.4 UDP/APPLICATION LAYER INTERFACE

The application interface to UDP MUST provide the full services of the IP/transport interface described in Section 3.4 of this

document. Thus, an application using UDP needs the functions of the GET_SRCADDR(), GET_MAXSIZES(), ADVISE_DELIVPROB(), and RECV_ICMP() calls described in Section 3.4. For example, GET_MAXSIZES() can be used to learn the effective maximum UDP maximum datagram size for a particular {interface,remote host,TOS} triplet.

An application-layer program MUST be able to set the TTL and TOS values as well as IP options for sending a UDP datagram, and these values must be passed transparently to the IP layer. UDP MAY pass the received TOS up to the application layer.

4.1.5 UDP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
UDP							
UDP send Port Unreachable	4.1.3.1		x				
IP Options in UDP							
- Pass rcv'd IP options to applic layer	4.1.3.2	x					
- Applic layer can specify IP options in Send	4.1.3.2	x					
- UDP passes IP options down to IP layer	4.1.3.2	x					
Pass ICMP msgs up to applic layer	4.1.3.3	x					
UDP checksums:							
- Able to generate/check checksum	4.1.3.4	x					
- Silently discard bad checksum	4.1.3.4	x					
- Sender Option to not generate checksum	4.1.3.4			x			
- Default is to checksum	4.1.3.4	x					
- Receiver Option to require checksum	4.1.3.4			x			
UDP Multihoming							
- Pass spec-dest addr to application	4.1.3.5	x					

- Applic layer can specify Local IP addr	4.1.3.5	x				
- Applic layer specify wild Local IP addr	4.1.3.5	x				
- Applic layer notified of Local IP addr used	4.1.3.5		x			
Bad IP src addr silently discarded by UDP/IP	4.1.3.6	x				
Only send valid IP source address	4.1.3.6	x				
UDP Application Interface Services						
Full IP interface of 3.4 for application	4.1.4	x				
- Able to spec TTL, TOS, IP opts when send dg	4.1.4	x				
- Pass received TOS up to applic layer	4.1.4			x		

4.2 TRANSMISSION CONTROL PROTOCOL -- TCP

4.2.1 INTRODUCTION

The Transmission Control Protocol TCP [TCP:1] is the primary virtual-circuit transport protocol for the Internet suite. TCP provides reliable, in-sequence delivery of a full-duplex stream of octets (8-bit bytes). TCP is used by those applications needing reliable, connection-oriented transport service, e.g., mail (SMTP), file transfer (FTP), and virtual terminal service (Telnet); requirements for these application-layer protocols are described in [INTRO:1].

4.2.2 PROTOCOL WALK-THROUGH

4.2.2.1 Well-Known Ports: RFC-793 Section 2.7

DISCUSSION:

TCP reserves port numbers in the range 0-255 for "well-known" ports, used to access services that are standardized across the Internet. The remainder of the port space can be freely allocated to application processes. Current well-known port definitions are listed in the RFC entitled "Assigned Numbers" [INTRO:6]. A prerequisite for defining a new well-known port is an RFC documenting the proposed service in enough detail to allow new implementations.

Some systems extend this notion by adding a third subdivision of the TCP port space: reserved ports, which are generally used for operating-system-specific services. For example, reserved ports might fall between 256 and some system-dependent upper limit. Some systems further choose to protect well-known and reserved ports by permitting only privileged users to open TCP connections with those port values. This is perfectly reasonable as long as the host does not assume that all hosts protect their low-numbered ports in this manner.

4.2.2.2 Use of Push: RFC-793 Section 2.8

When an application issues a series of SEND calls without setting the PUSH flag, the TCP MAY aggregate the data internally without sending it. Similarly, when a series of segments is received without the PSH bit, a TCP MAY queue the data internally without passing it to the receiving application.

The PSH bit is not a record marker and is independent of segment boundaries. The transmitter SHOULD collapse successive PSH bits when it packetizes data, to send the largest possible segment.

A TCP MAY implement PUSH flags on SEND calls. If PUSH flags are not implemented, then the sending TCP: (1) must not buffer data indefinitely, and (2) MUST set the PSH bit in the last buffered segment (i.e., when there is no more queued data to be sent).

The discussion in RFC-793 on pages 48, 50, and 74 erroneously implies that a received PSH flag must be passed to the application layer. Passing a received PSH flag to the application layer is now OPTIONAL.

An application program is logically required to set the PUSH flag in a SEND call whenever it needs to force delivery of the data to avoid a communication deadlock. However, a TCP SHOULD send a maximum-sized segment whenever possible, to improve performance (see Section 4.2.3.4).

DISCUSSION:

When the PUSH flag is not implemented on SEND calls, i.e., when the application/TCP interface uses a pure streaming model, responsibility for aggregating any tiny data fragments to form reasonable sized segments is partially borne by the application layer.

Generally, an interactive application protocol must set the PUSH flag at least in the last SEND call in each command or response sequence. A bulk transfer protocol like FTP should set the PUSH flag on the last segment of a file or when necessary to prevent buffer deadlock.

At the receiver, the PSH bit forces buffered data to be delivered to the application (even if less than a full buffer has been received). Conversely, the lack of a PSH bit can be used to avoid unnecessary wakeup calls to the application process; this can be an important performance optimization for large timesharing hosts. Passing the PSH bit to the receiving application allows an analogous optimization within the application.

4.2.2.3 Window Size: RFC-793 Section 3.1

The window size MUST be treated as an unsigned number, or else large window sizes will appear like negative windows

and TCP will not work. It is RECOMMENDED that implementations reserve 32-bit fields for the send and receive window sizes in the connection record and do all window computations with 32 bits.

DISCUSSION:
It is known that the window field in the TCP header is too small for high-speed, long-delay paths. Experimental TCP options have been defined to extend the window size; see for example [TCP:11]. In anticipation of the adoption of such an extension, TCP implementors should treat windows as 32 bits.

4.2.2.4 Urgent Pointer: RFC-793 Section 3.1

The second sentence is in error: the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data. The description on page 56 (last sentence) is correct.

A TCP MUST support a sequence of urgent data of any length.

A TCP MUST inform the application layer asynchronously whenever it receives an Urgent pointer and there was previously no pending urgent data, or whenever the Urgent pointer advances in the data stream. There MUST be a way for the application to learn how much urgent data remains to be read from the connection, or at least to determine whether or not more urgent data remains to be read.

DISCUSSION:
Although the Urgent mechanism may be used for any application, it is normally used to send "interrupt"-type commands to a Telnet program (see "Using Telnet Synch Sequence" section in [INTRO:1]).

The asynchronous or "out-of-band" notification will allow the application to go into "urgent mode", reading data from the TCP connection. This allows control commands to be sent to an application whose normal input buffers are full of unprocessed data.

IMPLEMENTATION:
The generic ERROR-REPORT() upcall described in Section 4.2.4.1 is a possible mechanism for informing the application of the arrival of urgent data.

4.2.2.5 TCP Options: RFC-793 Section 3.1

A TCP MUST be able to receive a TCP option in any segment. A TCP MUST ignore without error any TCP option it does not implement, assuming that the option has a length field (all TCP options defined in the future will have length fields). TCP MUST be prepared to handle an illegal option length (e.g., zero) without crashing; a suggested procedure is to reset the connection and log the reason.

4.2.2.6 Maximum Segment Size Option: RFC-793 Section 3.1

TCP MUST implement both sending and receiving the Maximum Segment Size option [TCP:4].

TCP SHOULD send an MSS (Maximum Segment Size) option in every SYN segment when its receive MSS differs from the default 536, and MAY send it always.

If an MSS option is not received at connection setup, TCP MUST assume a default send MSS of 536 (576-40) [TCP:4].

The maximum size of a segment that TCP really sends, the "effective send MSS," MUST be the smaller of the send MSS (which reflects the available reassembly buffer size at the remote host) and the largest size permitted by the IP layer:

```
Eff.snd.MSS =

    min(SendMSS+20, MMS_S) - TCPhdrsize - IPoptionsize
```

where:

* SendMSS is the MSS value received from the remote host, or the default 536 if no MSS option is received.

* MMS_S is the maximum size for a transport-layer message that TCP may send.

* TCPhdrsize is the size of the TCP header; this is normally 20, but may be larger if TCP options are to be sent.

* IPoptionsize is the size of any IP options that TCP will pass to the IP layer with the current message.

The MSS value to be sent in an MSS option must be less than

or equal to:

MMS_R - 20

where MMS_R is the maximum size for a transport-layer message that can be received (and reassembled). TCP obtains MMS_R and MMS_S from the IP layer; see the generic call GET_MAXSIZES in Section 3.4.

DISCUSSION:

The choice of TCP segment size has a strong effect on performance. Larger segments increase throughput by amortizing header size and per-datagram processing overhead over more data bytes; however, if the packet is so large that it causes IP fragmentation, efficiency drops sharply if any fragments are lost [IP:9].

Some TCP implementations send an MSS option only if the destination host is on a non-connected network. However, in general the TCP layer may not have the appropriate information to make this decision, so it is preferable to leave to the IP layer the task of determining a suitable MTU for the Internet path. We therefore recommend that TCP always send the option (if not 536) and that the IP layer determine MMS_R as specified in 3.3.3 and 3.4. A proposed IP-layer mechanism to measure the MTU would then modify the IP layer without changing TCP.

4.2.2.7 TCP Checksum: RFC-793 Section 3.1

Unlike the UDP checksum (see Section 4.1.3.4), the TCP checksum is never optional. The sender MUST generate it and the receiver MUST check it.

4.2.2.8 TCP Connection State Diagram: RFC-793 Section 3.2, page 23

There are several problems with this diagram:

(a) The arrow from SYN-SENT to SYN-RCVD should be labeled with "snd SYN,ACK", to agree with the text on page 68 and with Figure 8.

(b) There could be an arrow from SYN-RCVD state to LISTEN state, conditioned on receiving a RST after a passive open (see text page 70).

(c) It is possible to go directly from FIN-WAIT-1 to the TIME-WAIT state (see page 75 of the spec).

4.2.2.9 Initial Sequence Number Selection: RFC-793 Section 3.3, page 27

A TCP MUST use the specified clock-driven selection of initial sequence numbers.

4.2.2.10 Simultaneous Open Attempts: RFC-793 Section 3.4, page 32

There is an error in Figure 8: the packet on line 7 should be identical to the packet on line 5.

A TCP MUST support simultaneous open attempts.

DISCUSSION:
It sometimes surprises implementors that if two applications attempt to simultaneously connect to each other, only one connection is generated instead of two. This was an intentional design decision; don't try to "fix" it.

4.2.2.11 Recovery from Old Duplicate SYN: RFC-793 Section 3.4, page 33

Note that a TCP implementation MUST keep track of whether a connection has reached SYN_RCVD state as the result of a passive OPEN or an active OPEN.

4.2.2.12 RST Segment: RFC-793 Section 3.4

A TCP SHOULD allow a received RST segment to include data.

DISCUSSION
It has been suggested that a RST segment could contain ASCII text that encoded and explained the cause of the RST. No standard has yet been established for such data.

4.2.2.13 Closing a Connection: RFC-793 Section 3.5

A TCP connection may terminate in two ways: (1) the normal TCP close sequence using a FIN handshake, and (2) an "abort" in which one or more RST segments are sent and the connection state is immediately discarded. If a TCP

connection is closed by the remote site, the local application MUST be informed whether it closed normally or was aborted.

The normal TCP close sequence delivers buffered data reliably in both directions. Since the two directions of a TCP connection are closed independently, it is possible for a connection to be "half closed," i.e., closed in only one direction, and a host is permitted to continue sending data in the open direction on a half-closed connection.

A host MAY implement a "half-duplex" TCP close sequence, so that an application that has called CLOSE cannot continue to read data from the connection. If such a host issues a CLOSE call while received data is still pending in TCP, or if new data is received after CLOSE is called, its TCP SHOULD send a RST to show that data was lost.

When a connection is closed actively, it MUST linger in TIME-WAIT state for a time 2xMSL (Maximum Segment Lifetime). However, it MAY accept a new SYN from the remote TCP to reopen the connection directly from TIME-WAIT state, if it:

(1) assigns its initial sequence number for the new connection to be larger than the largest sequence number it used on the previous connection incarnation, and

(2) returns to TIME-WAIT state if the SYN turns out to be an old duplicate.

DISCUSSION:

TCP’s full-duplex data-preserving close is a feature that is not included in the analogous ISO transport protocol TP4.

Some systems have not implemented half-closed connections, presumably because they do not fit into the I/O model of their particular operating system. On these systems, once an application has called CLOSE, it can no longer read input data from the connection; this is referred to as a "half-duplex" TCP close sequence.

The graceful close algorithm of TCP requires that the connection state remain defined on (at least) one end of the connection, for a timeout period of 2xMSL, i.e., 4 minutes. During this period, the (remote socket,

local socket) pair that defines the connection is busy and cannot be reused. To shorten the time that a given port pair is tied up, some TCPs allow a new SYN to be accepted in TIME-WAIT state.

4.2.2.14 Data Communication: RFC-793 Section 3.7, page 40

Since RFC-793 was written, there has been extensive work on TCP algorithms to achieve efficient data communication. Later sections of the present document describe required and recommended TCP algorithms to determine when to send data (Section 4.2.3.4), when to send an acknowledgment (Section 4.2.3.2), and when to update the window (Section 4.2.3.3).

DISCUSSION:

One important performance issue is "Silly Window Syndrome" or "SWS" [TCP:5], a stable pattern of small incremental window movements resulting in extremely poor TCP performance. Algorithms to avoid SWS are described below for both the sending side (Section 4.2.3.4) and the receiving side (Section 4.2.3.3).

In brief, SWS is caused by the receiver advancing the right window edge whenever it has any new buffer space available to receive data and by the sender using any incremental window, no matter how small, to send more data [TCP:5]. The result can be a stable pattern of sending tiny data segments, even though both sender and receiver have a large total buffer space for the connection. SWS can only occur during the transmission of a large amount of data; if the connection goes quiescent, the problem will disappear. It is caused by typical straightforward implementation of window management, but the sender and receiver algorithms given below will avoid it.

Another important TCP performance issue is that some applications, especially remote login to character-at-a-time hosts, tend to send streams of one-octet data segments. To avoid deadlocks, every TCP SEND call from such applications must be "pushed", either explicitly by the application or else implicitly by TCP. The result may be a stream of TCP segments that contain one data octet each, which makes very inefficient use of the Internet and contributes to Internet congestion. The Nagle Algorithm described in Section 4.2.3.4 provides a simple and effective solution to this problem. It does have the effect of clumping

characters over Telnet connections; this may initially surprise users accustomed to single-character echo, but user acceptance has not been a problem.

Note that the Nagle algorithm and the send SWS avoidance algorithm play complementary roles in improving performance. The Nagle algorithm discourages sending tiny segments when the data to be sent increases in small increments, while the SWS avoidance algorithm discourages small segments resulting from the right window edge advancing in small increments.

A careless implementation can send two or more acknowledgment segments per data segment received. For example, suppose the receiver acknowledges every data segment immediately. When the application program subsequently consumes the data and increases the available receive buffer space again, the receiver may send a second acknowledgment segment to update the window at the sender. The extreme case occurs with single-character segments on TCP connections using the Telnet protocol for remote login service. Some implementations have been observed in which each incoming 1-character segment generates three return segments: (1) the acknowledgment, (2) a one byte increase in the window, and (3) the echoed character, respectively.

4.2.2.15 Retransmission Timeout: RFC-793 Section 3.7, page 41

The algorithm suggested in RFC-793 for calculating the retransmission timeout is now known to be inadequate; see Section 4.2.3.1 below.

Recent work by Jacobson [TCP:7] on Internet congestion and TCP retransmission stability has produced a transmission algorithm combining "slow start" with "congestion avoidance". A TCP MUST implement this algorithm.

If a retransmitted packet is identical to the original packet (which implies not only that the data boundaries have not changed, but also that the window and acknowledgment fields of the header have not changed), then the same IP Identification field MAY be used (see Section 3.2.1.5).

IMPLEMENTATION:
Some TCP implementors have chosen to "packetize" the data stream, i.e., to pick segment boundaries when

segments are originally sent and to queue these segments in a "retransmission queue" until they are acknowledged. Another design (which may be simpler) is to defer packetizing until each time data is transmitted or retransmitted, so there will be no segment retransmission queue.

In an implementation with a segment retransmission queue, TCP performance may be enhanced by repacketizing the segments awaiting acknowledgment when the first retransmission timeout occurs. That is, the outstanding segments that fitted would be combined into one maximum-sized segment, with a new IP Identification value. The TCP would then retain this combined segment in the retransmit queue until it was acknowledged. However, if the first two segments in the retransmission queue totalled more than one maximum-sized segment, the TCP would retransmit only the first segment using the original IP Identification field.

4.2.2.16 Managing the Window: RFC-793 Section 3.7, page 41

A TCP receiver SHOULD NOT shrink the window, i.e., move the right window edge to the left. However, a sending TCP MUST be robust against window shrinking, which may cause the "useable window" (see Section 4.2.3.4) to become negative.

If this happens, the sender SHOULD NOT send new data, but SHOULD retransmit normally the old unacknowledged data between SND.UNA and SND.UNA+SND.WND. The sender MAY also retransmit old data beyond SND.UNA+SND.WND, but SHOULD NOT time out the connection if data beyond the right window edge is not acknowledged. If the window shrinks to zero, the TCP MUST probe it in the standard way (see next Section).

DISCUSSION:

Many TCP implementations become confused if the window shrinks from the right after data has been sent into a larger window. Note that TCP has a heuristic to select the latest window update despite possible datagram reordering; as a result, it may ignore a window update with a smaller window than previously offered if neither the sequence number nor the acknowledgment number is increased.

4.2.2.17 Probing Zero Windows: RFC-793 Section 3.7, page 42

Probing of zero (offered) windows MUST be supported.

A TCP MAY keep its offered receive window closed indefinitely. As long as the receiving TCP continues to send acknowledgments in response to the probe segments, the sending TCP MUST allow the connection to stay open.

DISCUSSION:

It is extremely important to remember that ACK (acknowledgment) segments that contain no data are not reliably transmitted by TCP. If zero window probing is not supported, a connection may hang forever when an ACK segment that re-opens the window is lost.

The delay in opening a zero window generally occurs when the receiving application stops taking data from its TCP. For example, consider a printer daemon application, stopped because the printer ran out of paper.

The transmitting host SHOULD send the first zero-window probe when a zero window has existed for the retransmission timeout period (see Section 4.2.2.15), and SHOULD increase exponentially the interval between successive probes.

DISCUSSION:

This procedure minimizes delay if the zero-window condition is due to a lost ACK segment containing a window-opening update. Exponential backoff is recommended, possibly with some maximum interval not specified here. This procedure is similar to that of the retransmission algorithm, and it may be possible to combine the two procedures in the implementation.

4.2.2.18 Passive OPEN Calls: RFC-793 Section 3.8

Every passive OPEN call either creates a new connection record in LISTEN state, or it returns an error; it MUST NOT affect any previously created connection record.

A TCP that supports multiple concurrent users MUST provide an OPEN call that will functionally allow an application to LISTEN on a port while a connection block with the same local port is in SYN-SENT or SYN-RECEIVED state.

DISCUSSION:

Some applications (e.g., SMTP servers) may need to handle multiple connection attempts at about the same time. The probability of a connection attempt failing is reduced by giving the application some means of listening for a new connection at the same time that an earlier connection attempt is going through the three-way handshake.

IMPLEMENTATION:
Acceptable implementations of concurrent opens may permit multiple passive OPEN calls, or they may allow "cloning" of LISTEN-state connections from a single passive OPEN call.

4.2.2.19 Time to Live: RFC-793 Section 3.9, page 52

RFC-793 specified that TCP was to request the IP layer to send TCP segments with TTL = 60. This is obsolete; the TTL value used to send TCP segments MUST be configurable. See Section 3.2.1.7 for discussion.

4.2.2.20 Event Processing: RFC-793 Section 3.9

While it is not strictly required, a TCP SHOULD be capable of queueing out-of-order TCP segments. Change the "may" in the last sentence of the first paragraph on page 70 to "should".

DISCUSSION:
Some small-host implementations have omitted segment queueing because of limited buffer space. This omission may be expected to adversely affect TCP throughput, since loss of a single segment causes all later segments to appear to be "out of sequence".

In general, the processing of received segments MUST be implemented to aggregate ACK segments whenever possible. For example, if the TCP is processing a series of queued segments, it MUST process them all before sending any ACK segments.

Here are some detailed error corrections and notes on the Event Processing section of RFC-793.

(a) CLOSE Call, CLOSE-WAIT state, p. 61: enter LAST-ACK state, not CLOSING.

(b) LISTEN state, check for SYN (pp. 65, 66): With a SYN

bit, if the security/compartment or the precedence is wrong for the segment, a reset is sent. The wrong form of reset is shown in the text; it should be:

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

(c) SYN-SENT state, Check for SYN, p. 68: When the connection enters ESTABLISHED state, the following variables must be set:
SND.WND <- SEG.WND
SND.WL1 <- SEG.SEQ
SND.WL2 <- SEG.ACK

(d) Check security and precedence, p. 71: The first heading "ESTABLISHED STATE" should really be a list of all states other than SYN-RECEIVED: ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, and TIME-WAIT.

(e) Check SYN bit, p. 71: "In SYN-RECEIVED state and if the connection was initiated with a passive OPEN, then return this connection to the LISTEN state and return. Otherwise...".

(f) Check ACK field, SYN-RECEIVED state, p. 72: When the connection enters ESTABLISHED state, the variables listed in (c) must be set.

(g) Check ACK field, ESTABLISHED state, p. 72: The ACK is a duplicate if SEG.ACK =< SND.UNA (the = was omitted). Similarly, the window should be updated if: SND.UNA =< SEG.ACK =< SND.NXT.

(h) USER TIMEOUT, p. 77:

It would be better to notify the application of the timeout rather than letting TCP force the connection closed. However, see also Section 4.2.3.5.

4.2.2.21 Acknowledging Queued Segments: RFC-793 Section 3.9

A TCP MAY send an ACK segment acknowledging RCV.NXT when a valid segment arrives that is in the window but not at the left window edge.

DISCUSSION:

RFC-793 (see page 74) was ambiguous about whether or not an ACK segment should be sent when an out-of-order segment was received, i.e., when SEG.SEQ was unequal to RCV.NXT.

One reason for ACKing out-of-order segments might be to support an experimental algorithm known as "fast retransmit". With this algorithm, the sender uses the "redundant" ACK's to deduce that a segment has been lost before the retransmission timer has expired. It counts the number of times an ACK has been received with the same value of SEG.ACK and with the same right window edge. If more than a threshold number of such ACK's is received, then the segment containing the octets starting at SEG.ACK is assumed to have been lost and is retransmitted, without awaiting a timeout. The threshold is chosen to compensate for the maximum likely segment reordering in the Internet. There is not yet enough experience with the fast retransmit algorithm to determine how useful it is.

4.2.3 SPECIFIC ISSUES

4.2.3.1 Retransmission Timeout Calculation

A host TCP MUST implement Karn's algorithm and Jacobson's algorithm for computing the retransmission timeout ("RTO").

- Jacobson's algorithm for computing the smoothed round-trip ("RTT") time incorporates a simple measure of the variance [TCP:7].

- Karn's algorithm for selecting RTT measurements ensures that ambiguous round-trip times will not corrupt the calculation of the smoothed round-trip time [TCP:6].

This implementation also MUST include "exponential backoff" for successive RTO values for the same segment. Retransmission of SYN segments SHOULD use the same algorithm as data segments.

DISCUSSION:

There were two known problems with the RTO calculations specified in RFC-793. First, the accurate measurement of RTTs is difficult when there are retransmissions. Second, the algorithm to compute the smoothed round-trip time is inadequate [TCP:7], because it incorrectly

assumed that the variance in RTT values would be small and constant. These problems were solved by Karn's and Jacobson's algorithm, respectively.

The performance increase resulting from the use of these improvements varies from noticeable to dramatic. Jacobson's algorithm for incorporating the measured RTT variance is especially important on a low-speed link, where the natural variation of packet sizes causes a large variation in RTT. One vendor found link utilization on a 9.6kb line went from 10% to 90% as a result of implementing Jacobson's variance algorithm in TCP.

The following values SHOULD be used to initialize the estimation parameters for a new connection:

(a) RTT = 0 seconds.

(b) RTO = 3 seconds. (The smoothed variance is to be initialized to the value that will result in this RTO).

The recommended upper and lower bounds on the RTO are known to be inadequate on large internets. The lower bound SHOULD be measured in fractions of a second (to accommodate high speed LANs) and the upper bound should be 2*MSL, i.e., 240 seconds.

DISCUSSION:
Experience has shown that these initialization values are reasonable, and that in any case the Karn and Jacobson algorithms make TCP behavior reasonably insensitive to the initial parameter choices.

4.2.3.2 When to Send an ACK Segment

A host that is receiving a stream of TCP data segments can increase efficiency in both the Internet and the hosts by sending fewer than one ACK (acknowledgment) segment per data segment received; this is known as a "delayed ACK" [TCP:5].

A TCP SHOULD implement a delayed ACK, but an ACK should not be excessively delayed; in particular, the delay MUST be less than 0.5 seconds, and in a stream of full-sized segments there SHOULD be an ACK for at least every second segment.

DISCUSSION:

A delayed ACK gives the application an opportunity to update the window and perhaps to send an immediate response. In particular, in the case of character-mode remote login, a delayed ACK can reduce the number of segments sent by the server by a factor of 3 (ACK, window update, and echo character all combined in one segment).

In addition, on some large multi-user hosts, a delayed ACK can substantially reduce protocol processing overhead by reducing the total number of packets to be processed [TCP:5]. However, excessive delays on ACK's can disturb the round-trip timing and packet "clocking" algorithms [TCP:7].

4.2.3.3 When to Send a Window Update

A TCP MUST include a SWS avoidance algorithm in the receiver [TCP:5].

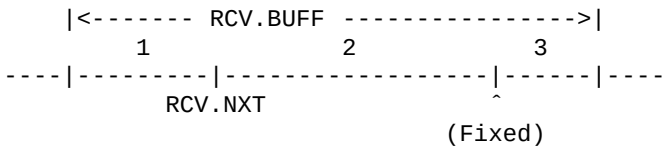
IMPLEMENTATION:

The receiver's SWS avoidance algorithm determines when the right window edge may be advanced; this is customarily known as "updating the window". This algorithm combines with the delayed ACK algorithm (see Section 4.2.3.2) to determine when an ACK segment containing the current window will really be sent to the receiver. We use the notation of RFC-793; see Figures 4 and 5 in that document.

The solution to receiver SWS is to avoid advancing the right window edge RCV.NXT+RCV.WND in small increments, even if data is received from the network in small segments.

Suppose the total receive buffer space is RCV.BUFF. At any given moment, RCV.USER octets of this total may be tied up with data that has been received and acknowledged but which the user process has not yet consumed. When the connection is quiescent, RCV.WND = RCV.BUFF and RCV.USER = 0.

Keeping the right window edge fixed as data arrives and is acknowledged requires that the receiver offer less than its full buffer space, i.e., the receiver must specify a RCV.WND that keeps RCV.NXT+RCV.WND constant as RCV.NXT increases. Thus, the total buffer space RCV.BUFF is generally divided into three parts:



1 - RCV.USER = data received but not yet consumed;
2 - RCV.WND = space advertised to sender;
3 - Reduction = space available but not yet advertised.

The suggested SWS avoidance algorithm for the receiver is to keep RCV.NXT+RCV.WND fixed until the reduction satisfies:

RCV.BUFF - RCV.USER - RCV.WND >=

min(Fr * RCV.BUFF, Eff.snd.MSS)

where Fr is a fraction whose recommended value is 1/2, and Eff.snd.MSS is the effective send MSS for the connection (see Section 4.2.2.6). When the inequality is satisfied, RCV.WND is set to RCV.BUFF-RCV.USER.

Note that the general effect of this algorithm is to advance RCV.WND in increments of Eff.snd.MSS (for realistic receive buffers: Eff.snd.MSS < RCV.BUFF/2). Note also that the receiver must use its own Eff.snd.MSS, assuming it is the same as the sender's.

4.2.3.4 When to Send Data

A TCP MUST include a SWS avoidance algorithm in the sender.

A TCP SHOULD implement the Nagle Algorithm [TCP:9] to coalesce short segments. However, there MUST be a way for an application to disable the Nagle algorithm on an individual connection. In all cases, sending data is also subject to the limitation imposed by the Slow Start algorithm (Section 4.2.2.15).

DISCUSSION:
The Nagle algorithm is generally as follows:

If there is unacknowledged data (i.e., SND.NXT > SND.UNA), then the sending TCP buffers all user

data (regardless of the PSH bit), until the

outstanding data has been acknowledged or until the TCP can send a full-sized segment (Eff.snd.MSS bytes; see Section 4.2.2.6).

Some applications (e.g., real-time display window updates) require that the Nagle algorithm be turned off, so small data segments can be streamed out at the maximum rate.

IMPLEMENTATION:

The sender's SWS avoidance algorithm is more difficult than the receivers's, because the sender does not know (directly) the receiver's total buffer space RCV.BUFF. An approach which has been found to work well is for the sender to calculate Max(SND.WND), the maximum send window it has seen so far on the connection, and to use this value as an estimate of RCV.BUFF. Unfortunately, this can only be an estimate; the receiver may at any time reduce the size of RCV.BUFF. To avoid a resulting deadlock, it is necessary to have a timeout to force transmission of data, overriding the SWS avoidance algorithm. In practice, this timeout should seldom occur.

The "useable window" [TCP:5] is:

U = SND.UNA + SND.WND - SND.NXT

i.e., the offered window less the amount of data sent but not acknowledged. If D is the amount of data queued in the sending TCP but not yet sent, then the following set of rules is recommended.

Send data:

(1) if a maximum-sized segment can be sent, i.e, if:

min(D,U) >= Eff.snd.MSS;

(2) or if the data is pushed and all queued data can be sent now, i.e., if:

[SND.NXT = SND.UNA and] PUSHED and D <= U

(the bracketed condition is imposed by the Nagle algorithm);

(3) or if at least a fraction Fs of the maximum window

can be sent, i.e., if:

[SND.NXT = SND.UNA and]

min(D.U) >= Fs * Max(SND.WND);

(4) or if data is PUSHed and the override timeout occurs.

Here Fs is a fraction whose recommended value is 1/2. The override timeout should be in the range 0.1 - 1.0 seconds. It may be convenient to combine this timer with the timer used to probe zero windows (Section 4.2.2.17).

Finally, note that the SWS avoidance algorithm just specified is to be used instead of the sender-side algorithm contained in [TCP:5].

4.2.3.5 TCP Connection Failures

Excessive retransmission of the same segment by TCP indicates some failure of the remote host or the Internet path. This failure may be of short or long duration. The following procedure MUST be used to handle excessive retransmissions of data segments [IP:11]:

(a) There are two thresholds R1 and R2 measuring the amount of retransmission that has occurred for the same segment. R1 and R2 might be measured in time units or as a count of retransmissions.

(b) When the number of transmissions of the same segment reaches or exceeds threshold R1, pass negative advice (see Section 3.3.1.4) to the IP layer, to trigger dead-gateway diagnosis.

(c) When the number of transmissions of the same segment reaches a threshold R2 greater than R1, close the connection.

(d) An application MUST be able to set the value for R2 for a particular connection. For example, an interactive application might set R2 to "infinity," giving the user control over when to disconnect.

(d) TCP SHOULD inform the application of the delivery

problem (unless such information has been disabled by the application; see Section 4.2.4.1), when R1 is reached and before R2. This will allow a remote login (User Telnet) application program to inform the user, for example.

The value of R1 SHOULD correspond to at least 3 retransmissions, at the current RTO. The value of R2 SHOULD correspond to at least 100 seconds.

An attempt to open a TCP connection could fail with excessive retransmissions of the SYN segment or by receipt of a RST segment or an ICMP Port Unreachable. SYN retransmissions MUST be handled in the general way just described for data retransmissions, including notification of the application layer.

However, the values of R1 and R2 may be different for SYN and data segments. In particular, R2 for a SYN segment MUST be set large enough to provide retransmission of the segment for at least 3 minutes. The application can close the connection (i.e., give up on the open attempt) sooner, of course.

DISCUSSION:
Some Internet paths have significant setup times, and the number of such paths is likely to increase in the future.

4.2.3.6 TCP Keep-Alives

Implementors MAY include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, the application MUST be able to turn them on or off for each TCP connection, and they MUST default to off.

Keep-alive packets MUST only be sent when no data or acknowledgement packets have been received for the connection within an interval. This interval MUST be configurable and MUST default to no less than two hours.

It is extremely important to remember that ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection.

An implementation SHOULD send a keep-alive segment with no

data; however, it MAY be configurable to send a keep-alive segment containing one garbage octet, for compatibility with erroneous TCP implementations.

DISCUSSION:

A "keep-alive" mechanism periodically probes the other end of a connection when the connection is otherwise idle, even when there is no data to be sent. The TCP specification does not include a keep-alive mechanism because it could: (1) cause perfectly good connections to break during transient Internet failures; (2) consume unnecessary bandwidth ("if no one is using the connection, who cares if it is still good?"); and (3) cost money for an Internet path that charges for packets.

Some TCP implementations, however, have included a keep-alive mechanism. To confirm that an idle connection is still active, these implementations send a probe segment designed to elicit a response from the peer TCP. Such a segment generally contains SEG.SEQ = SND.NXT-1 and may or may not contain one garbage octet of data. Note that on a quiet connection SND.NXT = RCV.NXT, so that this SEG.SEQ will be outside the window. Therefore, the probe causes the receiver to return an acknowledgment segment, confirming that the connection is still live. If the peer has dropped the connection due to a network partition or a crash, it will respond with a RST instead of an acknowledgment segment.

Unfortunately, some misbehaved TCP implementations fail to respond to a segment with SEG.SEQ = SND.NXT-1 unless the segment contains data. Alternatively, an implementation could determine whether a peer responded correctly to keep-alive packets with no garbage data octet.

A TCP keep-alive mechanism should only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure.

4.2.3.7 TCP Multihoming

If an application on a multihomed host does not specify the local IP address when actively opening a TCP connection, then the TCP MUST ask the IP layer to select a local IP address before sending the (first) SYN. See the function GET_SRCADDR() in Section 3.4.

At all other times, a previous segment has either been sent or received on this connection, and TCP MUST use the same local address is used that was used in those previous segments.

4.2.3.8 IP Options

When received options are passed up to TCP from the IP layer, TCP MUST ignore options that it does not understand.

A TCP MAY support the Time Stamp and Record Route options.

An application MUST be able to specify a source route when it actively opens a TCP connection, and this MUST take precedence over a source route received in a datagram.

When a TCP connection is OPENed passively and a packet arrives with a completed IP Source Route option (containing a return route), TCP MUST save the return route and use it for all segments sent on this connection. If a different source route arrives in a later segment, the later definition SHOULD override the earlier one.

4.2.3.9 ICMP Messages

TCP MUST act on an ICMP error message passed up from the IP layer, directing it to the connection that created the error. The necessary demultiplexing information can be found in the IP header contained within the ICMP message.

- Source Quench

  TCP MUST react to a Source Quench by slowing transmission on the connection. The RECOMMENDED procedure is for a Source Quench to trigger a "slow start," as if a retransmission timeout had occurred.

- Destination Unreachable -- codes 0, 1, 5

  Since these Unreachable messages indicate soft error

conditions, TCP MUST NOT abort the connection, and it SHOULD make the information available to the

application.

DISCUSSION:
TCP could report the soft error condition directly to the application layer with an upcall to the ERROR_REPORT routine, or it could merely note the message and report it to the application only when and if the TCP connection times out.

- o Destination Unreachable -- codes 2-4

  These are hard error conditions, so TCP SHOULD abort the connection.

- o Time Exceeded -- codes 0, 1

  This should be handled the same way as Destination Unreachable codes 0, 1, 5 (see above).

- o Parameter Problem

  This should be handled the same way as Destination Unreachable codes 0, 1, 5 (see above).

4.2.3.10 Remote Address Validation

A TCP implementation MUST reject as an error a local OPEN call for an invalid remote IP address (e.g., a broadcast or multicast address).

An incoming SYN with an invalid source address must be ignored either by TCP or by the IP layer (see Section 3.2.1.3).

A TCP implementation MUST silently discard an incoming SYN segment that is addressed to a broadcast or multicast address.

4.2.3.11 TCP Traffic Patterns

IMPLEMENTATION:
The TCP protocol specification [TCP:1] gives the implementor much freedom in designing the algorithms that control the message flow over the connection -- packetizing, managing the window, sending

acknowledgments, etc. These design decisions are difficult because a TCP must adapt to a wide range of

traffic patterns. Experience has shown that a TCP implementor needs to verify the design on two extreme traffic patterns:

o Single-character Segments

Even if the sender is using the Nagle Algorithm, when a TCP connection carries remote login traffic across a low-delay LAN the receiver will generally get a stream of single-character segments. If remote terminal echo mode is in effect, the receiver's system will generally echo each character as it is received.

o Bulk Transfer

When TCP is used for bulk transfer, the data stream should be made up (almost) entirely of segments of the size of the effective MSS. Although TCP uses a sequence number space with byte (octet) granularity, in bulk-transfer mode its operation should be as if TCP used a sequence space that counted only segments.

Experience has furthermore shown that a single TCP can effectively and efficiently handle these two extremes.

The most important tool for verifying a new TCP implementation is a packet trace program. There is a large volume of experience showing the importance of tracing a variety of traffic patterns with other TCP implementations and studying the results carefully.

4.2.3.12 Efficiency

IMPLEMENTATION:

Extensive experience has led to the following suggestions for efficient implementation of TCP:

(a) Don't Copy Data

In bulk data transfer, the primary CPU-intensive tasks are copying data from one place to another and checksumming the data. It is vital to minimize the number of copies of TCP data. Since

the ultimate speed limitation may be fetching data across the memory bus, it may be useful to combine

the copy with checksumming, doing both with a single memory fetch.

(b) Hand-Craft the Checksum Routine

A good TCP checksumming routine is typically two to five times faster than a simple and direct implementation of the definition. Great care and clever coding are often required and advisable to make the checksumming code "blazing fast". See [TCP:10].

(c) Code for the Common Case

TCP protocol processing can be complicated, but for most segments there are only a few simple decisions to be made. Per-segment processing will be greatly speeded up by coding the main line to minimize the number of decisions in the most common case.

4.2.4 TCP/APPLICATION LAYER INTERFACE

4.2.4.1 Asynchronous Reports

There MUST be a mechanism for reporting soft TCP error conditions to the application. Generically, we assume this takes the form of an application-supplied ERROR_REPORT routine that may be upcalled [INTRO:7] asynchronously from the transport layer:

ERROR_REPORT(local connection name, reason, subreason)

The precise encoding of the reason and subreason parameters is not specified here. However, the conditions that are reported asynchronously to the application MUST include:

* ICMP error message arrived (see 4.2.3.9)

* Excessive retransmissions (see 4.2.3.5)

* Urgent pointer advance (see 4.2.2.4).

However, an application program that does not want to receive such ERROR_REPORT calls SHOULD be able to

effectively disable these calls.

DISCUSSION:

These error reports generally reflect soft errors that can be ignored without harm by many applications. It has been suggested that these error report calls should default to "disabled," but this is not required.

4.2.4.2 Type-of-Service

The application layer MUST be able to specify the Type-of-Service (TOS) for segments that are sent on a connection. It not required, but the application SHOULD be able to change the TOS during the connection lifetime. TCP SHOULD pass the current TOS value without change to the IP layer, when it sends segments on the connection.

The TOS will be specified independently in each direction on the connection, so that the receiver application will specify the TOS used for ACK segments.

TCP MAY pass the most recently received TOS up to the application.

DISCUSSION

Some applications (e.g., SMTP) change the nature of their communication during the lifetime of a connection, and therefore would like to change the TOS specification.

Note also that the OPEN call specified in RFC-793 includes a parameter ("options") in which the caller can specify IP options such as source route, record route, or timestamp.

4.2.4.3 Flush Call

Some TCP implementations have included a FLUSH call, which will empty the TCP send queue of any data for which the user has issued SEND calls but which is still to the right of the current send window. That is, it flushes as much queued send data as possible without losing sequence number synchronization. This is useful for implementing the "abort output" function of Telnet.

4.2.4.4 Multihoming

The user interface outlined in sections 2.7 and 3.8 of RFC-793 needs to be extended for multihoming. The OPEN call MUST have an optional parameter:

OPEN(... [local IP address,] ...)

to allow the specification of the local IP address.

DISCUSSION:
Some TCP-based applications need to specify the local IP address to be used to open a particular connection; FTP is an example.

IMPLEMENTATION:
A passive OPEN call with a specified "local IP address" parameter will await an incoming connection request to that address. If the parameter is unspecified, a passive OPEN will await an incoming connection request to any local IP address, and then bind the local IP address of the connection to the particular address that is used.

For an active OPEN call, a specified "local IP address" parameter will be used for opening the connection. If the parameter is unspecified, the networking software will choose an appropriate local IP address (see Section 3.3.4.2) for the connection

4.2.5 TCP REQUIREMENT SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Conditional
Push flag							
Aggregate or queue un-pushed data	4.2.2.2			x			
Sender collapse successive PSH flags	4.2.2.2		x				
SEND call can specify PUSH	4.2.2.2			x			
If cannot: sender buffer indefinitely	4.2.2.2					x	
If cannot: PSH last segment	4.2.2.2	x					

Notify receiving ALP of PSH	4.2.2.2			x		1
Send max size segment when possible	4.2.2.2		x			
Window						
Treat as unsigned number	4.2.2.3	x				
Handle as 32-bit number	4.2.2.3		x			
Shrink window from right	4.2.2.16			x		
Robust against shrinking window	4.2.2.16	x				
Receiver's window closed indefinitely	4.2.2.17			x		
Sender probe zero window	4.2.2.17	x				
First probe after RTO	4.2.2.17		x			
Exponential backoff	4.2.2.17		x			
Allow window stay zero indefinitely	4.2.2.17	x				
Sender timeout OK conn with zero wind	4.2.2.17				x	
Urgent Data						
Pointer points to last octet	4.2.2.4	x				
Arbitrary length urgent data sequence	4.2.2.4	x				
Inform ALP asynchronously of urgent data	4.2.2.4	x				1
ALP can learn if/how much urgent data Q'd	4.2.2.4	x				1
TCP Options						
Receive TCP option in any segment	4.2.2.5	x				
Ignore unsupported options	4.2.2.5	x				
Cope with illegal option length	4.2.2.5	x				
Implement sending & receiving MSS option	4.2.2.6	x				
Send MSS option unless 536	4.2.2.6		x			
Send MSS option always	4.2.2.6			x		
Send-MSS default is 536	4.2.2.6	x				
Calculate effective send seg size	4.2.2.6	x				
TCP Checksums						
Sender compute checksum	4.2.2.7	x				
Receiver check checksum	4.2.2.7	x				
Use clock-driven ISN selection	4.2.2.9	x				
Opening Connections						
Support simultaneous open attempts	4.2.2.10	x				
SYN-RCVD remembers last state	4.2.2.11	x				
Passive Open call interfere with others	4.2.2.18				x	
Function: simultan. LISTENs for same port	4.2.2.18	x				
Ask IP for src address for SYN if necc.	4.2.3.7	x				
Otherwise, use local addr of conn.	4.2.3.7	x				
OPEN to broadcast/multicast IP Address	4.2.3.14				x	
Silently discard seg to bcast/mcast addr	4.2.3.14	x				

Closing Connections						

RST can contain data	4.2.2.12	x				
Inform application of aborted conn	4.2.2.13	x				
Half-duplex close connections	4.2.2.13		x			
Send RST to indicate data lost	4.2.2.13	x				
In TIME-WAIT state for 2xMSL seconds	4.2.2.13	x				
Accept SYN from TIME-WAIT state	4.2.2.13		x			
Retransmissions						
Jacobson Slow Start algorithm	4.2.2.15	x				
Jacobson Congestion-Avoidance algorithm	4.2.2.15	x				
Retransmit with same IP ident	4.2.2.15		x			
Karn's algorithm	4.2.3.1	x				
Jacobson's RTO estimation alg.	4.2.3.1	x				
Exponential backoff	4.2.3.1	x				
SYN RTO calc same as data	4.2.3.1		x			
Recommended initial values and bounds	4.2.3.1		x			
Generating ACK's:						
Queue out-of-order segments	4.2.2.20	x				
Process all Q'd before send ACK	4.2.2.20	x				
Send ACK for out-of-order segment	4.2.2.21		x			
Delayed ACK's	4.2.3.2		x			
Delay < 0.5 seconds	4.2.3.2	x				
Every 2nd full-sized segment ACK'd	4.2.3.2	x				
Receiver SWS-Avoidance Algorithm	4.2.3.3	x				
Sending data						
Configurable TTL	4.2.2.19	x				
Sender SWS-Avoidance Algorithm	4.2.3.4	x				
Nagle algorithm	4.2.3.4		x			
Application can disable Nagle algorithm	4.2.3.4	x				
Connection Failures:						
Negative advice to IP on R1 retxs	4.2.3.5	x				
Close connection on R2 retxs	4.2.3.5	x				
ALP can set R2	4.2.3.5	x				1
Inform ALP of R1<=retxs<R2	4.2.3.5		x			1
Recommended values for R1, R2	4.2.3.5		x			
Same mechanism for SYNs	4.2.3.5	x				
R2 at least 3 minutes for SYN	4.2.3.5	x				
Send Keep-alive Packets:	4.2.3.6			x		
- Application can request	4.2.3.6	x				
- Default is "off"	4.2.3.6	x				
- Only send if idle for interval	4.2.3.6	x				
- Interval configurable	4.2.3.6	x				

- Default at least 2 hrs.	4.2.3.6	x				
- Tolerant of lost ACK's	4.2.3.6	x				

IP Options							
Ignore options TCP doesn't understand	4.2.3.8	x					
Time Stamp support	4.2.3.8			x			
Record Route support	4.2.3.8			x			
Source Route:							
ALP can specify	4.2.3.8	x					1
Overrides src rt in datagram	4.2.3.8	x					
Build return route from src rt	4.2.3.8	x					
Later src route overrides	4.2.3.8		x				
Receiving ICMP Messages from IP	4.2.3.9	x					
Dest. Unreach (0,1,5) => inform ALP	4.2.3.9		x				
Dest. Unreach (0,1,5) => abort conn	4.2.3.9					x	
Dest. Unreach (2-4) => abort conn	4.2.3.9		x				
Source Quench => slow start	4.2.3.9		x				
Time Exceeded => tell ALP, don't abort	4.2.3.9		x				
Param Problem => tell ALP, don't abort	4.2.3.9		x				
Address Validation							
Reject OPEN call to invalid IP address	4.2.3.10	x					
Reject SYN from invalid IP address	4.2.3.10	x					
Silently discard SYN to bcast/mcast addr	4.2.3.10	x					
TCP/ALP Interface Services							
Error Report mechanism	4.2.4.1	x					
ALP can disable Error Report Routine	4.2.4.1		x				
ALP can specify TOS for sending	4.2.4.2	x					
Passed unchanged to IP	4.2.4.2		x				
ALP can change TOS during connection	4.2.4.2		x				
Pass received TOS up to ALP	4.2.4.2			x			
FLUSH call	4.2.4.3			x			
Optional local IP addr parm. in OPEN	4.2.4.4	x					

FOOTNOTES:

(1) "ALP" means Application-Layer program.

5. REFERENCES

INTRODUCTORY REFERENCES

[INTRO:1] "Requirements for Internet Hosts -- Application and Support," IETF Host Requirements Working Group, R. Braden, Ed., RFC-1123, October 1989.

[INTRO:2] "Requirements for Internet Gateways," R. Braden and J. Postel, RFC-1009, June 1987.

[INTRO:3] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:4] "Official Internet Protocols," J. Reynolds and J. Postel, RFC-1011, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:5] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC-980, March 1986.

[INTRO:6] "Assigned Numbers," J. Reynolds and J. Postel, RFC-1010, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:7] "Modularity and Efficiency in Protocol Implementations," D. Clark, RFC-817, July 1982.

[INTRO:8] "The Structuring of Systems Using Upcalls," D. Clark, 10th ACM SOSP, Orcas Island, Washington, December 1985.

Secondary References:

[INTRO:9] "A Protocol for Packet Network Intercommunication," V. Cerf and R. Kahn, IEEE Transactions on Communication, May 1974.

[INTRO:10] "The ARPA Internet Protocol," J. Postel, C. Sunshine, and D. Cohen, Computer Networks, Vol. 5, No. 4, July 1981.

[INTRO:11] "The DARPA Internet Protocol Suite," B. Leiner, J. Postel, R. Cole and D. Mills, Proceedings INFOCOM 85, IEEE, Washington DC,

March 1985. Also in: IEEE Communications Magazine, March 1985. Also available as ISI-RS-85-153.

[INTRO:12] "Final Text of DIS8473, Protocol for Providing the Connectionless Mode Network Service," ANSI, published as RFC-994, March 1986.

[INTRO:13] "End System to Intermediate System Routing Exchange Protocol," ANSI X3S3.3, published as RFC-995, April 1986.

LINK LAYER REFERENCES

[LINK:1] "Trailer Encapsulations," S. Leffler and M. Karels, RFC-893, April 1984.

[LINK:2] "An Ethernet Address Resolution Protocol," D. Plummer, RFC-826, November 1982.

[LINK:3] "A Standard for the Transmission of IP Datagrams over Ethernet Networks," C. Hornig, RFC-894, April 1984.

[LINK:4] "A Standard for the Transmission of IP Datagrams over IEEE 802 "Networks," J. Postel and J. Reynolds, RFC-1042, February 1988.

This RFC contains a great deal of information of importance to Internet implementers planning to use IEEE 802 networks.

IP LAYER REFERENCES

[IP:1] "Internet Protocol (IP)," J. Postel, RFC-791, September 1981.

[IP:2] "Internet Control Message Protocol (ICMP)," J. Postel, RFC-792, September 1981.

[IP:3] "Internet Standard Subnetting Procedure," J. Mogul and J. Postel, RFC-950, August 1985.

[IP:4] "Host Extensions for IP Multicasting," S. Deering, RFC-1112, August 1989.

[IP:5] "Military Standard Internet Protocol," MIL-STD-1777, Department of Defense, August 1983.

This specification, as amended by RFC-963, is intended to describe

the Internet Protocol but has some serious omissions (e.g., the mandatory subnet extension [IP:3] and the optional multicasting extension [IP:4]). It is also out of date. If there is a

conflict, RFC-791, RFC-792, and RFC-950 must be taken as authoritative, while the present document is authoritative over all.

[IP:6] "Some Problems with the Specification of the Military Standard Internet Protocol," D. Sidhu, RFC-963, November 1985.

[IP:7] "The TCP Maximum Segment Size and Related Topics," J. Postel, RFC-879, November 1983.

Discusses and clarifies the relationship between the TCP Maximum Segment Size option and the IP datagram size.

[IP:8] "Internet Protocol Security Options," B. Schofield, RFC-1108, October 1989.

[IP:9] "Fragmentation Considered Harmful," C. Kent and J. Mogul, ACM SIGCOMM-87, August 1987. Published as ACM Comp Comm Review, Vol. 17, no. 5.

This useful paper discusses the problems created by Internet fragmentation and presents alternative solutions.

[IP:10] "IP Datagram Reassembly Algorithms," D. Clark, RFC-815, July 1982.

This and the following paper should be read by every implementor.

[IP:11] "Fault Isolation and Recovery," D. Clark, RFC-816, July 1982.

SECONDARY IP REFERENCES:

[IP:12] "Broadcasting Internet Datagrams in the Presence of Subnets," J. Mogul, RFC-922, October 1984.

[IP:13] "Name, Addresses, Ports, and Routes," D. Clark, RFC-814, July 1982.

[IP:14] "Something a Host Could Do with Source Quench: The Source Quench Introduced Delay (SQUID)," W. Prue and J. Postel, RFC-1016, July 1987.

This RFC first described directed broadcast addresses. However, the bulk of the RFC is concerned with gateways, not hosts.

UDP REFERENCES:

[UDP:1] "User Datagram Protocol," J. Postel, RFC-768, August 1980.

TCP REFERENCES:

[TCP:1] "Transmission Control Protocol," J. Postel, RFC-793, September 1981.

[TCP:2] "Transmission Control Protocol," MIL-STD-1778, US Department of Defense, August 1984.

This specification as amended by RFC-964 is intended to describe the same protocol as RFC-793 [TCP:1]. If there is a conflict, RFC-793 takes precedence, and the present document is authoritative over both.

[TCP:3] "Some Problems with the Specification of the Military Standard Transmission Control Protocol," D. Sidhu and T. Blumer, RFC-964, November 1985.

[TCP:4] "The TCP Maximum Segment Size and Related Topics," J. Postel, RFC-879, November 1983.

[TCP:5] "Window and Acknowledgment Strategy in TCP," D. Clark, RFC-813, July 1982.

[TCP:6] "Round Trip Time Estimation," P. Karn & C. Partridge, ACM SIGCOMM-87, August 1987.

[TCP:7] "Congestion Avoidance and Control," V. Jacobson, ACM SIGCOMM-88, August 1988.

SECONDARY TCP REFERENCES:

[TCP:8] "Modularity and Efficiency in Protocol Implementation," D. Clark, RFC-817, July 1982.

[TCP:9] "Congestion Control in IP/TCP," J. Nagle, RFC-896, January 1984.

[TCP:10] "Computing the Internet Checksum," R. Braden, D. Borman, and C. Partridge, RFC-1071, September 1988.

[TCP:11] "TCP Extensions for Long-Delay Paths," V. Jacobson & R. Braden, RFC-1072, October 1988.

Security Considerations

There are many security issues in the communication layers of host software, but a full discussion is beyond the scope of this RFC.

The Internet architecture generally provides little protection against spoofing of IP source addresses, so any security mechanism that is based upon verifying the IP source address of a datagram should be treated with suspicion. However, in restricted environments some source-address checking may be possible. For example, there might be a secure LAN whose gateway to the rest of the Internet discarded any incoming datagram with a source address that spoofed the LAN address. In this case, a host on the LAN could use the source address to test for local vs. remote source. This problem is complicated by source routing, and some have suggested that source-routed datagram forwarding by hosts (see Section 3.3.5) should be outlawed for security reasons.

Security-related issues are mentioned in sections concerning the IP Security option (Section 3.2.1.8), the ICMP Parameter Problem message (Section 3.2.2.5), IP options in UDP datagrams (Section 4.1.3.2), and reserved TCP ports (Section 4.2.2.1).

Author's Address

Robert Braden
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (213) 822 1511

EMail: Braden@ISI.EDU

Appendix C

RFC-1123, Host Requirements - Application and Support

This appendix reproduces, in-full and un-edited, an RFC published by the Internet Engineering Task Force of the Internet Activities Board.

Network Working Group
Request for Comments: 1123

Internet Engineering Task Force
R. Braden, Editor

October 1989

Requirements for Internet Hosts -- Application and Support

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This RFC is one of a pair that defines and discusses the requirements for Internet host software. This RFC covers the application and support protocols; its companion RFC-1122 covers the communication protocol layers: link layer, IP layer, and transport layer.

Table of Contents

1. INTRODUCTION

This document is one of a pair that defines and discusses the requirements for host system implementations of the Internet protocol suite. This RFC covers the applications layer and support protocols. Its companion RFC, "Requirements for Internet Hosts -- Communications Layers" [INTRO:1] covers the lower layer protocols: transport layer, IP layer, and link layer.

These documents are intended to provide guidance for vendors, implementors, and users of Internet communication software. They represent the consensus of a large body of technical experience and wisdom, contributed by members of the Internet research and vendor communities.

This RFC enumerates standard protocols that a host connected to the Internet must use, and it incorporates by reference the RFCs and other documents describing the current specifications for these protocols. It corrects errors in the referenced documents and adds additional discussion and guidance for an implementor.

For each protocol, this document also contains an explicit set of requirements, recommendations, and options. The reader must understand that the list of requirements in this document is incomplete by itself; the complete set of requirements for an Internet host is primarily defined in the standard protocol specification documents, with the corrections, amendments, and supplements contained in this RFC.

A good-faith implementation of the protocols that was produced after careful reading of the RFC's and with some interaction with the Internet technical community, and that followed good communications software engineering practices, should differ from the requirements of this document in only minor ways. Thus, in many cases, the "requirements" in this RFC are already stated or implied in the standard protocol documents, so that their inclusion here is, in a sense, redundant. However, they were included because some past implementation has made the wrong choice, causing problems of interoperability, performance, and/or robustness.

This document includes discussion and explanation of many of the requirements and recommendations. A simple list of requirements would be dangerous, because:

- Some required features are more important than others, and some features are optional.

- There may be valid reasons why particular vendor products that

are designed for restricted contexts might choose to use different specifications.

However, the specifications of this document must be followed to meet the general goal of arbitrary host interoperation across the diversity and complexity of the Internet system. Although most current implementations fail to meet these requirements in various ways, some minor and some major, this specification is the ideal towards which we need to move.

These requirements are based on the current level of Internet architecture. This document will be updated as required to provide additional clarifications or to include additional information in those areas in which specifications are still evolving.

This introductory section begins with general advice to host software vendors, and then gives some guidance on reading the rest of the document. Section 2 contains general requirements that may be applicable to all application and support protocols. Sections 3, 4, and 5 contain the requirements on protocols for the three major applications: Telnet, file transfer, and electronic mail, respectively. Section 6 covers the support applications: the domain name system, system initialization, and management. Finally, all references will be found in Section 7.

1.1 The Internet Architecture

For a brief introduction to the Internet architecture from a host viewpoint, see Section 1.1 of [INTRO:1]. That section also contains recommended references for general background on the Internet architecture.

1.2 General Considerations

There are two important lessons that vendors of Internet host software have learned and which a new vendor should consider seriously.

1.2.1 Continuing Internet Evolution

The enormous growth of the Internet has revealed problems of management and scaling in a large datagram-based packet communication system. These problems are being addressed, and as a result there will be continuing evolution of the specifications described in this document. These changes will be carefully planned and controlled, since there is extensive participation in this planning by the vendors and by the organizations responsible for operations of the networks.

Development, evolution, and revision are characteristic of computer network protocols today, and this situation will persist for some years. A vendor who develops computer communication software for the Internet protocol suite (or any other protocol suite!) and then fails to maintain and update that software for changing specifications is going to leave a trail of unhappy customers. The Internet is a large communication network, and the users are in constant contact through it. Experience has shown that knowledge of deficiencies in vendor software propagates quickly through the Internet technical community.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability:

> "Be liberal in what you accept, and
> conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field -- e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up. An undefined code might be logged (see below), but it must not cause a failure.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is "watch out

for misbehaving hosts"; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

1.2.3 Error Logging

The Internet includes a great variety of host and gateway systems, each implementing many protocols and protocol layers, and some of these contain bugs and mis-features in their Internet protocol software. As a result of complexity, diversity, and distribution of function, the diagnosis of user problems is often very difficult.

Problem diagnosis will be aided if host implementations include a carefully designed facility for logging erroneous or "strange" protocol events. It is important to include as much diagnostic information as possible when an error is logged. In particular, it is often useful to record the header(s) of a packet that caused an error. However, care must be taken to ensure that error logging does not consume prohibitive amounts of resources or otherwise interfere with the operation of the host.

There is a tendency for abnormal but harmless protocol events to overflow error logging files; this can be avoided by using a "circular" log, or by enabling logging only while diagnosing a known failure. It may be useful to filter and count duplicate successive messages. One strategy that seems to work well is: (1) always count abnormalities and make such counts accessible through the management protocol (see Section 6.3); and (2) allow the logging of a great variety of events to be selectively enabled. For example, it might useful to be able to "log everything" or to "log everything for host X".

Note that different managements may have differing policies about the amount of error logging that they want normally enabled in a host. Some will say, "if it doesn’t hurt me, I don’t want to know about it", while others will want to take a more watchful and aggressive attitude about detecting and removing protocol abnormalities.

1.2.4 Configuration

It would be ideal if a host implementation of the Internet protocol suite could be entirely self-configuring. This would allow the whole suite to be implemented in ROM or cast into silicon, it would simplify diskless workstations, and it would

be an immense boon to harried LAN administrators as well as system vendors. We have not reached this ideal; in fact, we are not even close.

At many points in this document, you will find a requirement that a parameter be a configurable option. There are several different reasons behind such requirements. In a few cases, there is current uncertainty or disagreement about the best value, and it may be necessary to update the recommended value in the future. In other cases, the value really depends on external factors -- e.g., the size of the host and the distribution of its communication load, or the speeds and topology of nearby networks -- and self-tuning algorithms are unavailable and may be insufficient. In some cases, configurability is needed because of administrative requirements.

Finally, some configuration options are required to communicate with obsolete or incorrect implementations of the protocols, distributed without sources, that unfortunately persist in many parts of the Internet. To make correct systems coexist with these faulty systems, administrators often have to "mis-configure" the correct systems. This problem will correct itself gradually as the faulty systems are retired, but it cannot be ignored by vendors.

When we say that a parameter must be configurable, we do not intend to require that its value be explicitly read from a configuration file at every boot time. We recommend that implementors set up a default for each parameter, so a configuration file is only necessary to override those defaults that are inappropriate in a particular installation. Thus, the configurability requirement is an assurance that it will be POSSIBLE to override the default when necessary, even in a binary-only or ROM-based product.

This document requires a particular value for such defaults in some cases. The choice of default is a sensitive issue when the configuration item controls the accommodation to existing faulty systems. If the Internet is to converge successfully to complete interoperability, the default values built into implementations must implement the official protocol, not "mis-configurations" to accommodate faulty implementations. Although marketing considerations have led some vendors to choose mis-configuration defaults, we urge vendors to choose defaults that will conform to the standard.

Finally, we note that a vendor needs to provide adequate

documentation on all configuration parameters, their limits and effects.

1.3 Reading this Document

1.3.1 Organization

In general, each major section is organized into the following subsections:

(1) Introduction

(2) Protocol Walk-Through -- considers the protocol specification documents section-by-section, correcting errors, stating requirements that may be ambiguous or ill-defined, and providing further clarification or explanation.

(3) Specific Issues -- discusses protocol design and implementation issues that were not included in the walk-through.

(4) Interfaces -- discusses the service interface to the next higher layer.

(5) Summary -- contains a summary of the requirements of the section.

Under many of the individual topics in this document, there is parenthetical material labeled "DISCUSSION" or "IMPLEMENTATION". This material is intended to give clarification and explanation of the preceding requirements text. It also includes some suggestions on possible future directions or developments. The implementation material contains suggested approaches that an implementor may want to consider.

The summary sections are intended to be guides and indexes to the text, but are necessarily cryptic and incomplete. The summaries should never be used or referenced separately from the complete RFC.

1.3.2 Requirements

In this document, the words that are used to define the significance of each particular requirement are capitalized. These words are:

* "MUST"

  This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

* "SHOULD"

  This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant".

1.3.3 Terminology

This document uses the following technical terms:

Segment
: A segment is the unit of end-to-end transmission in the TCP protocol. A segment consists of a TCP header followed by application data. A segment is transmitted by encapsulation in an IP datagram.

Message
: This term is used by some application layer protocols (particularly SMTP) for an application data unit.

Datagram
: A [UDP] datagram is the unit of end-to-end transmission in the UDP protocol.

Multihomed
A host is said to be multihomed if it has multiple IP addresses to connected networks.

1.4 Acknowledgments

This document incorporates contributions and comments from a large group of Internet protocol experts, including representatives of university and research labs, vendors, and government agencies. It was assembled primarily by the Host Requirements Working Group of the Internet Engineering Task Force (IETF).

The Editor would especially like to acknowledge the tireless dedication of the following people, who attended many long meetings and generated 3 million bytes of electronic mail over the past 18 months in pursuit of this document: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), and James Van Bokkelen (FTP Software).

In addition, the following people made major contributions to the effort: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), and Mike StJohns (DCA). The following also made significant contributions to particular areas: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), and Rayan Zachariassen (Toronto).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

2. GENERAL ISSUES

This section contains general requirements that may be applicable to all application-layer protocols.

2.1 Host Names and Numbers

The syntax of a legal Internet host name was specified in RFC-952 [DNS:4]. One aspect of host name syntax is hereby changed: the restriction on the first character is relaxed to allow either a letter or a digit. Host software MUST support this more liberal syntax.

Host software MUST handle host names of up to 63 characters and SHOULD handle host names of up to 255 characters.

Whenever a user inputs the identity of an Internet host, it SHOULD be possible to enter either (1) a host domain name or (2) an IP address in dotted-decimal ("#.#.#.#") form. The host SHOULD check the string syntactically for a dotted-decimal number before looking it up in the Domain Name System.

DISCUSSION:

This last requirement is not intended to specify the complete syntactic form for entering a dotted-decimal host number; that is considered to be a user-interface issue. For example, a dotted-decimal number must be enclosed within "[ ]" brackets for SMTP mail (see Section 5.2.17). This notation could be made universal within a host system, simplifying the syntactic checking for a dotted-decimal number.

If a dotted-decimal number can be entered without such identifying delimiters, then a full syntactic check must be made, because a segment of a host domain name is now allowed to begin with a digit and could legally be entirely numeric (see Section 6.1.2.4). However, a valid host name can never have the dotted-decimal form #.#.#.#, since at least the highest-level component label will be alphabetic.

2.2 Using Domain Name Service

Host domain names MUST be translated to IP addresses as described in Section 6.1.

Applications using domain name services MUST be able to cope with soft error conditions. Applications MUST wait a reasonable interval between successive retries due to a soft error, and MUST

allow for the possibility that network problems may deny service for hours or even days.

An application SHOULD NOT rely on the ability to locate a WKS record containing an accurate listing of all services at a particular host address, since the WKS RR type is not often used by Internet sites. To confirm that a service is present, simply attempt to use it.

2.3 Applications on Multihomed hosts

When the remote host is multihomed, the name-to-address translation will return a list of alternative IP addresses. As specified in Section 6.1.3.4, this list should be in order of decreasing preference. Application protocol implementations SHOULD be prepared to try multiple addresses from the list until success is obtained. More specific requirements for SMTP are given in Section 5.3.4.

When the local host is multihomed, a UDP-based request/response application SHOULD send the response with an IP source address that is the same as the specific destination address of the UDP request datagram. The "specific destination address" is defined in the "IP Addressing" section of the companion RFC [INTRO:1].

Similarly, a server application that opens multiple TCP connections to the same client SHOULD use the same local IP address for all.

2.4 Type-of-Service

Applications MUST select appropriate TOS values when they invoke transport layer services, and these values MUST be configurable. Note that a TOS value contains 5 bits, of which only the most-significant 3 bits are currently defined; the other two bits MUST be zero.

DISCUSSION:

As gateway algorithms are developed to implement Type-of-Service, the recommended values for various application protocols may change. In addition, it is likely that particular combinations of users and Internet paths will want non-standard TOS values. For these reasons, the TOS values must be configurable.

See the latest version of the "Assigned Numbers" RFC [INTRO:5] for the recommended TOS values for the major application protocols.

2.5 GENERAL APPLICATION REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MUST NOT	SHOULD NOT	MAY	Footnote
User interfaces:							
Allow host name to begin with digit	2.1	x					
Host names of up to 635 characters	2.1	x					
Host names of up to 255 characters	2.1		x				
Support dotted-decimal host numbers	2.1		x				
Check syntactically for dotted-dec first	2.1		x				
Map domain names per Section 6.1	2.2	x					
Cope with soft DNS errors	2.2	x					
Reasonable interval between retries	2.2	x					
Allow for long outages	2.2	x					
Expect WKS records to be available	2.2				x		
Try multiple addr's for remote multihomed host	2.3		x				
UDP reply src addr is specific dest of request	2.3		x				
Use same IP addr for related TCP connections	2.3		x				
Specify appropriate TOS values	2.4	x					
TOS values configurable	2.4	x					
Unused TOS bits zero	2.4	x					

3. REMOTE LOGIN -- TELNET PROTOCOL

3.1 INTRODUCTION

Telnet is the standard Internet application protocol for remote login. It provides the encoding rules to link a user's keyboard/display on a client ("user") system with a command interpreter on a remote server system. A subset of the Telnet protocol is also incorporated within other application protocols, e.g., FTP and SMTP.

Telnet uses a single TCP connection, and its normal data stream ("Network Virtual Terminal" or "NVT" mode) is 7-bit ASCII with escape sequences to embed control functions. Telnet also allows the negotiation of many optional modes and functions.

The primary Telnet specification is to be found in RFC-854 [TELNET:1], while the options are defined in many other RFCs; see Section 7 for references.

3.2 PROTOCOL WALK-THROUGH

3.2.1 Option Negotiation: RFC-854, pp. 2-3

Every Telnet implementation MUST include option negotiation and subnegotiation machinery [TELNET:2].

A host MUST carefully follow the rules of RFC-854 to avoid option-negotiation loops. A host MUST refuse (i.e, reply WONT/DONT to a DO/WILL) an unsupported option. Option negotiation SHOULD continue to function (even if all requests are refused) throughout the lifetime of a Telnet connection.

If all option negotiations fail, a Telnet implementation MUST default to, and support, an NVT.

DISCUSSION:
Even though more sophisticated "terminals" and supporting option negotiations are becoming the norm, all implementations must be prepared to support an NVT for any user-server communication.

3.2.2 Telnet Go-Ahead Function: RFC-854, p. 5, and RFC-858

On a host that never sends the Telnet command Go Ahead (GA), the Telnet Server MUST attempt to negotiate the Suppress Go Ahead option (i.e., send "WILL Suppress Go Ahead"). A User or Server Telnet MUST always accept negotiation of the Suppress Go

Ahead option.

When it is driving a full-duplex terminal for which GA has no meaning, a User Telnet implementation MAY ignore GA commands.

DISCUSSION:

Half-duplex ("locked-keyboard") line-at-a-time terminals for which the Go-Ahead mechanism was designed have largely disappeared from the scene. It turned out to be difficult to implement sending the Go-Ahead signal in many operating systems, even some systems that support native half-duplex terminals. The difficulty is typically that the Telnet server code does not have access to information about whether the user process is blocked awaiting input from the Telnet connection, i.e., it cannot reliably determine when to send a GA command. Therefore, most Telnet Server hosts do not send GA commands.

The effect of the rules in this section is to allow either end of a Telnet connection to veto the use of GA commands.

There is a class of half-duplex terminals that is still commercially important: "data entry terminals," which interact in a full-screen manner. However, supporting data entry terminals using the Telnet protocol does not require the Go Ahead signal; see Section 3.3.2.

3.2.3 Control Functions: RFC-854, pp. 7-8

The list of Telnet commands has been extended to include EOR (End-of-Record), with code 239 [TELNET:9].

Both User and Server Telnets MAY support the control functions EOR, EC, EL, and Break, and MUST support AO, AYT, DM, IP, NOP, SB, and SE.

A host MUST be able to receive and ignore any Telnet control functions that it does not support.

DISCUSSION:

Note that a Server Telnet is required to support the Telnet IP (Interrupt Process) function, even if the server host has an equivalent in-stream function (e.g., Control-C in many systems). The Telnet IP function may be stronger than an in-stream interrupt command, because of the out-of-band effect of TCP urgent data.

The EOR control function may be used to delimit the

stream. An important application is data entry terminal support (see Section 3.3.2). There was concern that since EOR had not been defined in RFC-854, a host that was not prepared to correctly ignore unknown Telnet commands might crash if it received an EOR. To protect such hosts, the End-of-Record option [TELNET:9] was introduced; however, a properly implemented Telnet program will not require this protection.

3.2.4 Telnet "Synch" Signal: RFC-854, pp. 8-10

When it receives "urgent" TCP data, a User or Server Telnet MUST discard all data except Telnet commands until the DM (and end of urgent) is reached.

When it sends Telnet IP (Interrupt Process), a User Telnet SHOULD follow it by the Telnet "Synch" sequence, i.e., send as TCP urgent data the sequence "IAC IP IAC DM". The TCP urgent pointer points to the DM octet.

When it receives a Telnet IP command, a Server Telnet MAY send a Telnet "Synch" sequence back to the user, to flush the output stream. The choice ought to be consistent with the way the server operating system behaves when a local user interrupts a process.

When it receives a Telnet AO command, a Server Telnet MUST send a Telnet "Synch" sequence back to the user, to flush the output stream.

A User Telnet SHOULD have the capability of flushing output when it sends a Telnet IP; see also Section 3.4.5.

DISCUSSION:

There are three possible ways for a User Telnet to flush the stream of server output data:

(1) Send AO after IP.

This will cause the server host to send a "flush-buffered-output" signal to its operating system. However, the AO may not take effect locally, i.e., stop terminal output at the User Telnet end, until the Server Telnet has received and processed the AO and has sent back a "Synch".

(2) Send DO TIMING-MARK [TELNET:7] after IP, and discard all output locally until a WILL/WONT TIMING-MARK is

received from the Server Telnet.

Since the DO TIMING-MARK will be processed after the IP at the server, the reply to it should be in the right place in the output data stream. However, the TIMING-MARK will not send a "flush buffered output" signal to the server operating system. Whether or not this is needed is dependent upon the server system.

(3) Do both.

The best method is not entirely clear, since it must accommodate a number of existing server hosts that do not follow the Telnet standards in various ways. The safest approach is probably to provide a user-controllable option to select (1), (2), or (3).

3.2.5 NVT Printer and Keyboard: RFC-854, p. 11

In NVT mode, a Telnet SHOULD NOT send characters with the high-order bit 1, and MUST NOT send it as a parity bit. Implementations that pass the high-order bit to applications SHOULD negotiate binary mode (see Section 3.2.6).

DISCUSSION:

Implementors should be aware that a strict reading of RFC-854 allows a client or server expecting NVT ASCII to ignore characters with the high-order bit set. In general, binary mode is expected to be used for transmission of an extended (beyond 7-bit) character set with Telnet.

However, there exist applications that really need an 8-bit NVT mode, which is currently not defined, and these existing applications do set the high-order bit during part or all of the life of a Telnet connection. Note that binary mode is not the same as 8-bit NVT mode, since binary mode turns off end-of-line processing. For this reason, the requirements on the high-order bit are stated as SHOULD, not MUST.

RFC-854 defines a minimal set of properties of a "network virtual terminal" or NVT; this is not meant to preclude additional features in a real terminal. A Telnet connection is fully transparent to all 7-bit ASCII characters, including arbitrary ASCII control characters.

For example, a terminal might support full-screen commands coded as ASCII escape sequences; a Telnet implementation would pass these sequences as uninterpreted data. Thus, an NVT should not be conceived as a terminal type of a highly-restricted device.

3.2.6 Telnet Command Structure: RFC-854, p. 13

Since options may appear at any point in the data stream, a Telnet escape character (known as IAC, with the value 255) to be sent as data MUST be doubled.

3.2.7 Telnet Binary Option: RFC-856

When the Binary option has been successfully negotiated, arbitrary 8-bit characters are allowed. However, the data stream MUST still be scanned for IAC characters, any embedded Telnet commands MUST be obeyed, and data bytes equal to IAC MUST be doubled. Other character processing (e.g., replacing CR by CR NUL or by CR LF) MUST NOT be done. In particular, there is no end-of-line convention (see Section 3.3.1) in binary mode.

DISCUSSION:

The Binary option is normally negotiated in both directions, to change the Telnet connection from NVT mode to "binary mode".

The sequence IAC EOR can be used to delimit blocks of data within a binary-mode Telnet stream.

3.2.8 Telnet Terminal-Type Option: RFC-1091

The Terminal-Type option MUST use the terminal type names officially defined in the Assigned Numbers RFC [INTRO:5], when they are available for the particular terminal. However, the receiver of a Terminal-Type option MUST accept any name.

DISCUSSION:

RFC-1091 [TELNET:10] updates an earlier version of the Terminal-Type option defined in RFC-930. The earlier version allowed a server host capable of supporting multiple terminal types to learn the type of a particular client's terminal, assuming that each physical terminal had an intrinsic type. However, today a "terminal" is often really a terminal emulator program running in a PC, perhaps capable of emulating a range of terminal types. Therefore, RFC-1091 extends the specification to allow a

more general terminal-type negotiation between User and Server Telnets.

3.3 SPECIFIC ISSUES

3.3.1 Telnet End-of-Line Convention

The Telnet protocol defines the sequence CR LF to mean "end-of-line". For terminal input, this corresponds to a command-completion or "end-of-line" key being pressed on a user terminal; on an ASCII terminal, this is the CR key, but it may also be labelled "Return" or "Enter".

When a Server Telnet receives the Telnet end-of-line sequence CR LF as input from a remote terminal, the effect MUST be the same as if the user had pressed the "end-of-line" key on a local terminal. On server hosts that use ASCII, in particular, receipt of the Telnet sequence CR LF must cause the same effect as a local user pressing the CR key on a local terminal. Thus, CR LF and CR NUL MUST have the same effect on an ASCII server host when received as input over a Telnet connection.

A User Telnet MUST be able to send any of the forms: CR LF, CR NUL, and LF. A User Telnet on an ASCII host SHOULD have a user-controllable mode to send either CR LF or CR NUL when the user presses the "end-of-line" key, and CR LF SHOULD be the default.

The Telnet end-of-line sequence CR LF MUST be used to send Telnet data that is not terminal-to-computer (e.g., for Server Telnet sending output, or the Telnet protocol incorporated another application protocol).

DISCUSSION:

To allow interoperability between arbitrary Telnet clients and servers, the Telnet protocol defined a standard representation for a line terminator. Since the ASCII character set includes no explicit end-of-line character, systems have chosen various representations, e.g., CR, LF, and the sequence CR LF. The Telnet protocol chose the CR LF sequence as the standard for network transmission.

Unfortunately, the Telnet protocol specification in RFC-854 [TELNET:1] has turned out to be somewhat ambiguous on what character(s) should be sent from client to server for the "end-of-line" key. The result has been a massive and continuing interoperability headache, made worse by various faulty implementations of both User and Server

Telnets.

Although the Telnet protocol is based on a perfectly symmetric model, in a remote login session the role of the user at a terminal differs from the role of the server host. For example, RFC-854 defines the meaning of CR, LF, and CR LF as output from the server, but does not specify what the User Telnet should send when the user presses the "end-of-line" key on the terminal; this turns out to be the point at issue.

When a user presses the "end-of-line" key, some User Telnet implementations send CR LF, while others send CR NUL (based on a different interpretation of the same sentence in RFC-854). These will be equivalent for a correctly-implemented ASCII server host, as discussed above. For other servers, a mode in the User Telnet is needed.

The existence of User Telnets that send only CR NUL when CR is pressed creates a dilemma for non-ASCII hosts: they can either treat CR NUL as equivalent to CR LF in input, thus precluding the possibility of entering a "bare" CR, or else lose complete interworking.

Suppose a user on host A uses Telnet to log into a server host B, and then execute B's User Telnet program to log into server host C. It is desirable for the Server/User Telnet combination on B to be as transparent as possible, i.e., to appear as if A were connected directly to C. In particular, correct implementation will make B transparent to Telnet end-of-line sequences, except that CR LF may be translated to CR NUL or vice versa.

IMPLEMENTATION:

To understand Telnet end-of-line issues, one must have at least a general model of the relationship of Telnet to the local operating system. The Server Telnet process is typically coupled into the terminal driver software of the operating system as a pseudo-terminal. A Telnet end-of-line sequence received by the Server Telnet must have the same effect as pressing the end-of-line key on a real locally-connected terminal.

Operating systems that support interactive character-at-a-time applications (e.g., editors) typically have two internal modes for their terminal I/O: a formatted mode, in which local conventions for end-of-line and other

formatting rules have been applied to the data stream, and a "raw" mode, in which the application has direct access to every character as it was entered. A Server Telnet must be implemented in such a way that these modes have the same effect for remote as for local terminals. For example, suppose a CR LF or CR NUL is received by the Server Telnet on an ASCII host. In raw mode, a CR character is passed to the application; in formatted mode, the local system's end-of-line convention is used.

3.3.2 Data Entry Terminals

DISCUSSION:

In addition to the line-oriented and character-oriented ASCII terminals for which Telnet was designed, there are several families of video display terminals that are sometimes known as "data entry terminals" or DETs. The IBM 3270 family is a well-known example.

Two Internet protocols have been designed to support generic DETs: SUPDUP [TELNET:16, TELNET:17], and the DET option [TELNET:18, TELNET:19]. The DET option drives a data entry terminal over a Telnet connection using (sub-) negotiation. SUPDUP is a completely separate terminal protocol, which can be entered from Telnet by negotiation. Although both SUPDUP and the DET option have been used successfully in particular environments, neither has gained general acceptance or wide implementation.

A different approach to DET interaction has been developed for supporting the IBM 3270 family through Telnet, although the same approach would be applicable to any DET. The idea is to enter a "native DET" mode, in which the native DET input/output stream is sent as binary data. The Telnet EOR command is used to delimit logical records (e.g., "screens") within this binary stream.

IMPLEMENTATION:

The rules for entering and leaving native DET mode are as follows:

- The Server uses the Terminal-Type option [TELNET:10] to learn that the client is a DET.
- It is conventional, but not required, that both ends negotiate the EOR option [TELNET:9].
- Both ends negotiate the Binary option [TELNET:3] to

enter native DET mode.

- When either end negotiates out of binary mode, the other end does too, and the mode then reverts to normal NVT.

3.3.3 Option Requirements

Every Telnet implementation MUST support the Binary option [TELNET:3] and the Suppress Go Ahead option [TELNET:5], and SHOULD support the Echo [TELNET:4], Status [TELNET:6], End-of-Record [TELNET:9], and Extended Options List [TELNET:8] options.

A User or Server Telnet SHOULD support the Window Size Option [TELNET:12] if the local operating system provides the corresponding capability.

DISCUSSION:
Note that the End-of-Record option only signifies that a Telnet can receive a Telnet EOR without crashing; therefore, every Telnet ought to be willing to accept negotiation of the End-of-Record option. See also the discussion in Section 3.2.3.

3.3.4 Option Initiation

When the Telnet protocol is used in a client/server situation, the server SHOULD initiate negotiation of the terminal interaction mode it expects.

DISCUSSION:
The Telnet protocol was defined to be perfectly symmetrical, but its application is generally asymmetric. Remote login has been known to fail because NEITHER side initiated negotiation of the required non-default terminal modes. It is generally the server that determines the preferred mode, so the server needs to initiate the negotiation; since the negotiation is symmetric, the user can also initiate it.

A client (User Telnet) SHOULD provide a means for users to enable and disable the initiation of option negotiation.

DISCUSSION:
A user sometimes needs to connect to an application service (e.g., FTP or SMTP) that uses Telnet for its

control stream but does not support Telnet options. User Telnet may be used for this purpose if initiation of option negotiation is disabled.

3.3.5 Telnet Linemode Option

DISCUSSION:

An important new Telnet option, LINEMODE [TELNET:12], has been proposed. The LINEMODE option provides a standard way for a User Telnet and a Server Telnet to agree that the client rather than the server will perform terminal character processing. When the client has prepared a complete line of text, it will send it to the server in (usually) one TCP packet. This option will greatly decrease the packet cost of Telnet sessions and will also give much better user response over congested or long-delay networks.

The LINEMODE option allows dynamic switching between local and remote character processing. For example, the Telnet connection will automatically negotiate into single-character mode while a full screen editor is running, and then return to linemode when the editor is finished.

We expect that when this RFC is released, hosts should implement the client side of this option, and may implement the server side of this option. To properly implement the server side, the server needs to be able to tell the local system not to do any input character processing, but to remember its current terminal state and notify the Server Telnet process whenever the state changes. This will allow password echoing and full screen editors to be handled properly, for example.

3.4 TELNET/USER INTERFACE

3.4.1 Character Set Transparency

User Telnet implementations SHOULD be able to send or receive any 7-bit ASCII character. Where possible, any special character interpretations by the user host’s operating system SHOULD be bypassed so that these characters can conveniently be sent and received on the connection.

Some character value MUST be reserved as "escape to command mode"; conventionally, doubling this character allows it to be entered as data. The specific character used SHOULD be user selectable.

On binary-mode connections, a User Telnet program MAY provide an escape mechanism for entering arbitrary 8-bit values, if the host operating system doesn't allow them to be entered directly from the keyboard.

IMPLEMENTATION:

The transparency issues are less pressing on servers, but implementors should take care in dealing with issues like: masking off parity bits (sent by an older, non-conforming client) before they reach programs that expect only NVT ASCII, and properly handling programs that request 8-bit data streams.

3.4.2 Telnet Commands

A User Telnet program MUST provide a user the capability of entering any of the Telnet control functions IP, AO, or AYT, and SHOULD provide the capability of entering EC, EL, and Break.

3.4.3 TCP Connection Errors

A User Telnet program SHOULD report to the user any TCP errors that are reported by the transport layer (see "TCP/Application Layer Interface" section in [INTRO:1]).

3.4.4 Non-Default Telnet Contact Port

A User Telnet program SHOULD allow the user to optionally specify a non-standard contact port number at the Server Telnet host.

3.4.5 Flushing Output

A User Telnet program SHOULD provide the user the ability to specify whether or not output should be flushed when an IP is sent; see Section 3.2.4.

For any output flushing scheme that causes the User Telnet to flush output locally until a Telnet signal is received from the Server, there SHOULD be a way for the user to manually restore normal output, in case the Server fails to send the expected signal.

3.5. TELNET REQUIREMENTS SUMMARY

FEATURE	SECTION	SHOULD	SHOULD NOT	MUST	MUST NOT	OPTIONAL
Option Negotiation	3.2.1	x				
Avoid negotiation loops	3.2.1	x				
Refuse unsupported options	3.2.1	x				
Negotiation OK anytime on connection	3.2.1		x			
Default to NVT	3.2.1	x				
Send official name in Term-Type option	3.2.8	x				
Accept any name in Term-Type option	3.2.8	x				
Implement Binary, Suppress-GA options	3.3.3	x				
Echo, Status, EOL, Ext-Opt-List options	3.3.3		x			
Implement Window-Size option if appropriate	3.3.3		x			
Server initiate mode negotiations	3.3.4		x			
User can enable/disable init negotiations	3.3.4		x			
Go-Aheads						
Non-GA server negotiate SUPPRESS-GA option	3.2.2	x				
User or Server accept SUPPRESS-GA option	3.2.2	x				
User Telnet ignore GA’s	3.2.2			x		
Control Functions						
Support SE NOP DM IP AO AYT SB	3.2.3	x				
Support EOR EC EL Break	3.2.3			x		
Ignore unsupported control functions	3.2.3	x				
User, Server discard urgent data up to DM	3.2.4	x				
User Telnet send "Synch" after IP, AO, AYT	3.2.4		x			
Server Telnet reply Synch to IP	3.2.4			x		
Server Telnet reply Synch to AO	3.2.4	x				
User Telnet can flush output when send IP	3.2.4		x			
Encoding						
Send high-order bit in NVT mode	3.2.5				x	
Send high-order bit as parity bit	3.2.5					x
Negot. BINARY if pass high-ord. bit to applic	3.2.5		x			
Always double IAC data byte	3.2.6	x				

Double IAC data byte in binary mode	3.2.7	x				
Obey Telnet cmds in binary mode	3.2.7	x				
End-of-line, CR NUL in binary mode	3.2.7					x
End-of-Line						
EOL at Server same as local end-of-line	3.3.1	x				
ASCII Server accept CR LF or CR NUL for EOL	3.3.1	x				
User Telnet able to send CR LF, CR NUL, or LF	3.3.1	x				
ASCII user able to select CR LF/CR NUL	3.3.1		x			
User Telnet default mode is CR LF	3.3.1		x			
Non-interactive uses CR LF for EOL	3.3.1	x				
User Telnet interface						
Input & output all 7-bit characters	3.4.1		x			
Bypass local op sys interpretation	3.4.1		x			
Escape character	3.4.1	x				
User-settable escape character	3.4.1		x			
Escape to enter 8-bit values	3.4.1			x		
Can input IP, AO, AYT	3.4.2	x				
Can input EC, EL, Break	3.4.2		x			
Report TCP connection errors to user	3.4.3		x			
Optional non-default contact port	3.4.4		x			
Can spec: output flushed when IP sent	3.4.5		x			
Can manually restore output mode	3.4.5		x			

4. FILE TRANSFER

4.1 FILE TRANSFER PROTOCOL -- FTP

4.1.1 INTRODUCTION

The File Transfer Protocol FTP is the primary Internet standard for file transfer. The current specification is contained in RFC-959 [FTP:1].

FTP uses separate simultaneous TCP connections for control and for data transfer. The FTP protocol includes many features, some of which are not commonly implemented. However, for every feature in FTP, there exists at least one implementation. The minimum implementation defined in RFC-959 was too small, so a somewhat larger minimum implementation is defined here.

Internet users have been unnecessarily burdened for years by deficient FTP implementations. Protocol implementors have suffered from the erroneous opinion that implementing FTP ought to be a small and trivial task. This is wrong, because FTP has a user interface, because it has to deal (correctly) with the whole variety of communication and operating system errors that may occur, and because it has to handle the great diversity of real file systems in the world.

4.1.2. PROTOCOL WALK-THROUGH

4.1.2.1 LOCAL Type: RFC-959 Section 3.1.1.4

An FTP program MUST support TYPE I ("IMAGE" or binary type) as well as TYPE L 8 ("LOCAL" type with logical byte size 8). A machine whose memory is organized into m-bit words, where m is not a multiple of 8, MAY also support TYPE L m.

DISCUSSION:

The command "TYPE L 8" is often required to transfer binary data between a machine whose memory is organized into (e.g.) 36-bit words and a machine with an 8-bit byte organization. For an 8-bit byte machine, TYPE L 8 is equivalent to IMAGE.

"TYPE L m" is sometimes specified to the FTP programs on two m-bit word machines to ensure the correct transfer of a native-mode binary file from one machine to the other. However, this command should have the same effect on these machines as "TYPE I".

4.1.2.2 Telnet Format Control: RFC-959 Section 3.1.1.5.2

A host that makes no distinction between TYPE N and TYPE T SHOULD implement TYPE T to be identical to TYPE N.

DISCUSSION:

This provision should ease interoperation with hosts that do make this distinction.

Many hosts represent text files internally as strings of ASCII characters, using the embedded ASCII format effector characters (LF, BS, FF, ...) to control the format when a file is printed. For such hosts, there is no distinction between "print" files and other files. However, systems that use record structured files typically need a special format for printable files (e.g., ASA carriage control). For the latter hosts, FTP allows a choice of TYPE N or TYPE T.

4.1.2.3 Page Structure: RFC-959 Section 3.1.2.3 and Appendix I

Implementation of page structure is NOT RECOMMENDED in general. However, if a host system does need to implement FTP for "random access" or "holey" files, it MUST use the defined page structure format rather than define a new private FTP format.

4.1.2.4 Data Structure Transformations: RFC-959 Section 3.1.2

An FTP transformation between record-structure and file-structure SHOULD be invertible, to the extent possible while making the result useful on the target host.

DISCUSSION:

RFC-959 required strict invertibility between record-structure and file-structure, but in practice, efficiency and convenience often preclude it. Therefore, the requirement is being relaxed. There are two different objectives for transferring a file: processing it on the target host, or just storage. For storage, strict invertibility is important. For processing, the file created on the target host needs to be in the format expected by application programs on that host.

As an example of the conflict, imagine a record-oriented operating system that requires some data files to have exactly 80 bytes in each record. While STORing

a file on such a host, an FTP Server must be able to pad each line or record to 80 bytes; a later retrieval of such a file cannot be strictly invertible.

4.1.2.5 Data Connection Management: RFC-959 Section 3.3

A User-FTP that uses STREAM mode SHOULD send a PORT command to assign a non-default data port before each transfer command is issued.

DISCUSSION:

This is required because of the long delay after a TCP connection is closed until its socket pair can be reused, to allow multiple transfers during a single FTP session. Sending a port command can avoided if a transfer mode other than stream is used, by leaving the data transfer connection open between transfers.

4.1.2.6 PASV Command: RFC-959 Section 4.1.2

A server-FTP MUST implement the PASV command.

If multiple third-party transfers are to be executed during the same session, a new PASV command MUST be issued before each transfer command, to obtain a unique port pair.

IMPLEMENTATION:

The format of the 227 reply to a PASV command is not well standardized. In particular, an FTP client cannot assume that the parentheses shown on page 40 of RFC-959 will be present (and in fact, Figure 3 on page 43 omits them). Therefore, a User-FTP program that interprets the PASV reply must scan the reply for the first digit of the host and port numbers.

Note that the host number h1,h2,h3,h4 is the IP address of the server host that is sending the reply, and that p1,p2 is a non-default data transfer port that PASV has assigned.

4.1.2.7 LIST and NLST Commands: RFC-959 Section 4.1.3

The data returned by an NLST command MUST contain only a simple list of legal pathnames, such that the server can use them directly as the arguments of subsequent data transfer commands for the individual files.

The data returned by a LIST or NLST command SHOULD use an

implied TYPE AN, unless the current type is EBCDIC, in which case an implied TYPE EN SHOULD be used.

DISCUSSION:

Many FTP clients support macro-commands that will get or put files matching a wildcard specification, using NLST to obtain a list of pathnames. The expansion of "multiple-put" is local to the client, but "multiple-get" requires cooperation by the server.

The implied type for LIST and NLST is designed to provide compatibility with existing User-FTPs, and in particular with multiple-get commands.

4.1.2.8 SITE Command: RFC-959 Section 4.1.3

A Server-FTP SHOULD use the SITE command for non-standard features, rather than invent new private commands or unstandardized extensions to existing commands.

4.1.2.9 STOU Command: RFC-959 Section 4.1.3

The STOU command stores into a uniquely named file. When it receives an STOU command, a Server-FTP MUST return the actual file name in the "125 Transfer Starting" or the "150 Opening Data Connection" message that precedes the transfer (the 250 reply code mentioned in RFC-959 is incorrect). The exact format of these messages is hereby defined to be as follows:

```
125 FILE: pppp
150 FILE: pppp
```

where pppp represents the unique pathname of the file that will be written.

4.1.2.10 Telnet End-of-line Code: RFC-959, Page 34

Implementors MUST NOT assume any correspondence between READ boundaries on the control connection and the Telnet EOL sequences (CR LF).

DISCUSSION:

Thus, a server-FTP (or User-FTP) must continue reading characters from the control connection until a complete Telnet EOL sequence is encountered, before processing the command (or response, respectively). Conversely, a single READ from the control connection may include

more than one FTP command.

4.1.2.11 FTP Replies: RFC-959 Section 4.2, Page 35

A Server-FTP MUST send only correctly formatted replies on the control connection. Note that RFC-959 (unlike earlier versions of the FTP spec) contains no provision for a "spontaneous" reply message.

A Server-FTP SHOULD use the reply codes defined in RFC-959 whenever they apply. However, a server-FTP MAY use a different reply code when needed, as long as the general rules of Section 4.2 are followed. When the implementor has a choice between a 4xx and 5xx reply code, a Server-FTP SHOULD send a 4xx (temporary failure) code when there is any reasonable possibility that a failed FTP will succeed a few hours later.

A User-FTP SHOULD generally use only the highest-order digit of a 3-digit reply code for making a procedural decision, to prevent difficulties when a Server-FTP uses non-standard reply codes.

A User-FTP MUST be able to handle multi-line replies. If the implementation imposes a limit on the number of lines and if this limit is exceeded, the User-FTP MUST recover, e.g., by ignoring the excess lines until the end of the multi-line reply is reached.

A User-FTP SHOULD NOT interpret a 421 reply code ("Service not available, closing control connection") specially, but SHOULD detect closing of the control connection by the server.

DISCUSSION:

Server implementations that fail to strictly follow the reply rules often cause FTP user programs to hang. Note that RFC-959 resolved ambiguities in the reply rules found in earlier FTP specifications and must be followed.

It is important to choose FTP reply codes that properly distinguish between temporary and permanent failures, to allow the successful use of file transfer client daemons. These programs depend on the reply codes to decide whether or not to retry a failed transfer; using a permanent failure code (5xx) for a temporary error will cause these programs to give up unnecessarily.

When the meaning of a reply matches exactly the text shown in RFC-959, uniformity will be enhanced by using the RFC-959 text verbatim. However, a Server-FTP implementor is encouraged to choose reply text that conveys specific system-dependent information, when appropriate.

4.1.2.12 Connections: RFC-959 Section 5.2

The words "and the port used" in the second paragraph of this section of RFC-959 are erroneous (historical), and they should be ignored.

On a multihomed server host, the default data transfer port (L-1) MUST be associated with the same local IP address as the corresponding control connection to port L.

A user-FTP MUST NOT send any Telnet controls other than SYNCH and IP on an FTP control connection. In particular, it MUST NOT attempt to negotiate Telnet options on the control connection. However, a server-FTP MUST be capable of accepting and refusing Telnet negotiations (i.e., sending DONT/WONT).

DISCUSSION:

Although the RFC says: "Server- and User- processes should follow the conventions for the Telnet protocol...[on the control connection]", it is not the intent that Telnet option negotiation is to be employed.

4.1.2.13 Minimum Implementation; RFC-959 Section 5.1

The following commands and options MUST be supported by every server-FTP and user-FTP, except in cases where the underlying file system or operating system does not allow or support a particular command.

Type: ASCII Non-print, IMAGE, LOCAL 8
Mode: Stream
Structure: File, Record*
Commands:
USER, PASS, ACCT,
PORT, PASV,
TYPE, MODE, STRU,
RETR, STOR, APPE,
RNFR, RNTO, DELE,
CWD, CDUP, RMD, MKD, PWD,

LIST, NLST,
SYST, STAT,
HELP, NOOP, QUIT.

*Record structure is REQUIRED only for hosts whose file systems support record structure.

DISCUSSION:
Vendors are encouraged to implement a larger subset of the protocol. For example, there are important robustness features in the protocol (e.g., Restart, ABOR, block mode) that would be an aid to some Internet users but are not widely implemented.

A host that does not have record structures in its file system may still accept files with STRU R, recording the byte stream literally.

4.1.3 SPECIFIC ISSUES

4.1.3.1 Non-standard Command Verbs

FTP allows "experimental" commands, whose names begin with "X". If these commands are subsequently adopted as standards, there may still be existing implementations using the "X" form. At present, this is true for the directory commands:

RFC-959	"Experimental"
MKD	XMKD
RMD	XRMD
PWD	XPWD
CDUP	XCUP
CWD	XCWD

All FTP implementations SHOULD recognize both forms of these commands, by simply equating them with extra entries in the command lookup table.

IMPLEMENTATION:
A User-FTP can access a server that supports only the "X" forms by implementing a mode switch, or automatically using the following procedure: if the RFC-959 form of one of the above commands is rejected with a 500 or 502 response code, then try the experimental form; any other response would be passed to the user.

4.1.3.2 Idle Timeout

A Server-FTP process SHOULD have an idle timeout, which will terminate the process and close the control connection if the server is inactive (i.e., no command or data transfer in progress) for a long period of time. The idle timeout time SHOULD be configurable, and the default should be at least 5 minutes.

A client FTP process ("User-PI" in RFC-959) will need timeouts on responses only if it is invoked from a program.

DISCUSSION:

Without a timeout, a Server-FTP process may be left pending indefinitely if the corresponding client crashes without closing the control connection.

4.1.3.3 Concurrency of Data and Control

DISCUSSION:

The intent of the designers of FTP was that a user should be able to send a STAT command at any time while data transfer was in progress and that the server-FTP would reply immediately with status -- e.g., the number of bytes transferred so far. Similarly, an ABOR command should be possible at any time during a data transfer.

Unfortunately, some small-machine operating systems make such concurrent programming difficult, and some other implementers seek minimal solutions, so some FTP implementations do not allow concurrent use of the data and control connections. Even such a minimal server must be prepared to accept and defer a STAT or ABOR command that arrives during data transfer.

4.1.3.4 FTP Restart Mechanism

The description of the 110 reply on pp. 40-41 of RFC-959 is incorrect; the correct description is as follows. A restart reply message, sent over the control connection from the receiving FTP to the User-FTP, has the format:

```
110 MARK ssss = rrrr
```

Here:

* ssss is a text string that appeared in a Restart Marker

in the data stream and encodes a position in the sender's file system;

* rrrr encodes the corresponding position in the receiver's file system.

The encoding, which is specific to a particular file system and network implementation, is always generated and interpreted by the same system, either sender or receiver.

When an FTP that implements restart receives a Restart Marker in the data stream, it SHOULD force the data to that point to be written to stable storage before encoding the corresponding position rrrr. An FTP sending Restart Markers MUST NOT assume that 110 replies will be returned synchronously with the data, i.e., it must not await a 110 reply before sending more data.

Two new reply codes are hereby defined for errors encountered in restarting a transfer:

554 Requested action not taken: invalid REST parameter.

A 554 reply may result from a FTP service command that follows a REST command. The reply indicates that the existing file at the Server-FTP cannot be repositioned as specified in the REST.

555 Requested action not taken: type or stru mismatch.

A 555 reply may result from an APPE command or from any FTP service command following a REST command. The reply indicates that there is some mismatch between the current transfer parameters (type and stru) and the attributes of the existing file.

DISCUSSION:

Note that the FTP Restart mechanism requires that Block or Compressed mode be used for data transfer, to allow the Restart Markers to be included within the data stream. The frequency of Restart Markers can be low.

Restart Markers mark a place in the data stream, but the receiver may be performing some transformation on the data as it is stored into stable storage. In general, the receiver's encoding must include any state information necessary to restart this transformation at any point of the FTP data stream. For example, in TYPE

A transfers, some receiver hosts transform CR LF sequences into a single LF character on disk. If a Restart Marker happens to fall between CR and LF, the receiver must encode in rrrr that the transfer must be restarted in a "CR has been seen and discarded" state.

Note that the Restart Marker is required to be encoded as a string of printable ASCII characters, regardless of the type of the data.

RFC-959 says that restart information is to be returned "to the user". This should not be taken literally. In general, the User-FTP should save the restart information (ssss,rrrr) in stable storage, e.g., append it to a restart control file. An empty restart control file should be created when the transfer first starts and deleted automatically when the transfer completes successfully. It is suggested that this file have a name derived in an easily-identifiable manner from the name of the file being transferred and the remote host name; this is analogous to the means used by many text editors for naming "backup" files.

There are three cases for FTP restart.

(1) User-to-Server Transfer

The User-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the Server-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and transformation state as rrrr, and returns a "110 MARK ssss = rrrr" reply over the control connection. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, repositions its local file system and transformation state using ssss, and sends the command "REST rrrr" to the Server-FTP.

(2) Server-to-User Transfer

The Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the User-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and

transformation state as rrrr, and appends the pair (rrrr,ssss) to its restart control file.

To restart the transfer, the User-FTP fetches the last (rrrr,ssss) pair from the restart control file, repositions its local file system and transformation state using rrrr, and sends the command "REST ssss" to the Server-FTP.

(3) Server-to-Server ("Third-Party") Transfer

The sending Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When it receives a Marker, the receiving Server-FTP writes all prior data to disk, encodes its file system position and transformation state as rrrr, and sends a "110 MARK ssss = rrrr" reply over the control connection to the User. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, sends "REST ssss" to the sending Server-FTP, and sends "REST rrrr" to the receiving Server-FTP.

4.1.4 FTP/USER INTERFACE

This section discusses the user interface for a User-FTP program.

4.1.4.1 Pathname Specification

Since FTP is intended for use in a heterogeneous environment, User-FTP implementations MUST support remote pathnames as arbitrary character strings, so that their form and content are not limited by the conventions of the local operating system.

DISCUSSION:
In particular, remote pathnames can be of arbitrary length, and all the printing ASCII characters as well as space (0x20) must be allowed. RFC-959 allows a pathname to contain any 7-bit ASCII character except CR or LF.

4.1.4.2 "QUOTE" Command

A User-FTP program MUST implement a "QUOTE" command that will pass an arbitrary character string to the server and display all resulting response messages to the user.

To make the "QUOTE" command useful, a User-FTP SHOULD send transfer control commands to the server as the user enters them, rather than saving all the commands and sending them to the server only when a data transfer is started.

DISCUSSION:

The "QUOTE" command is essential to allow the user to access servers that require system-specific commands (e.g., SITE or ALLO), or to invoke new or optional features that are not implemented by the User-FTP. For example, "QUOTE" may be used to specify "TYPE A T" to send a print file to hosts that require the distinction, even if the User-FTP does not recognize that TYPE.

4.1.4.3 Displaying Replies to User

A User-FTP SHOULD display to the user the full text of all error reply messages it receives. It SHOULD have a "verbose" mode in which all commands it sends and the full text and reply codes it receives are displayed, for diagnosis of problems.

4.1.4.4 Maintaining Synchronization

The state machine in a User-FTP SHOULD be forgiving of missing and unexpected reply messages, in order to maintain command synchronization with the server.

4.1.5 FTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHLD	MAY	NOT	FUTURE
Implement TYPE T if same as TYPE N	4.1.2.2		x			
File/Record transform invertible if poss.	4.1.2.4		x			
User-FTP send PORT cmd for stream mode	4.1.2.5		x			
Server-FTP implement PASV	4.1.2.6	x				
PASV is per-transfer	4.1.2.6	x				
NLST reply usable in RETR cmds	4.1.2.7	x				
Implied type for LIST and NLST	4.1.2.7		x			
SITE cmd for non-standard features	4.1.2.8		x			
STOU cmd return pathname as specified	4.1.2.9	x				
Use TCP READ boundaries on control conn.	4.1.2.10				x	
Server-FTP send only correct reply format	4.1.2.11	x				
Server-FTP use defined reply code if poss.	4.1.2.11		x			
New reply code following Section 4.2	4.1.2.11			x		
User-FTP use only high digit of reply	4.1.2.11		x			
User-FTP handle multi-line reply lines	4.1.2.11	x				
User-FTP handle 421 reply specially	4.1.2.11			x		
Default data port same IP addr as ctl conn	4.1.2.12	x				
User-FTP send Telnet cmds exc. SYNCH, IP	4.1.2.12				x	
User-FTP negotiate Telnet options	4.1.2.12				x	
Server-FTP handle Telnet options	4.1.2.12	x				
Handle "Experimental" directory cmds	4.1.3.1		x			
Idle timeout in server-FTP	4.1.3.2		x			
Configurable idle timeout	4.1.3.2		x			
Receiver checkpoint data at Restart Marker	4.1.3.4		x			
Sender assume 110 replies are synchronous	4.1.3.4				x	
Support TYPE:						
ASCII - Non-Print (AN)	4.1.2.13	x				
ASCII - Telnet (AT) -- if same as AN	4.1.2.2		x			
ASCII - Carriage Control (AC)	959 3.1.1.5.2			x		
EBCDIC - (any form)	959 3.1.1.2			x		
IMAGE	4.1.2.1	x				
LOCAL 8	4.1.2.1	x				

LOCAL m	4.1.2.1			x		2
Support MODE:						
Stream	4.1.2.13	x				
Block	959 3.4.2			x		
Support STRUCTURE:						
File	4.1.2.13	x				
Record	4.1.2.13	x				3
Page	4.1.2.3				x	
Support commands:						
USER	4.1.2.13	x				
PASS	4.1.2.13	x				
ACCT	4.1.2.13	x				
CWD	4.1.2.13	x				
CDUP	4.1.2.13	x				
SMNT	959 5.3.1			x		
REIN	959 5.3.1			x		
QUIT	4.1.2.13	x				
PORT	4.1.2.13	x				
PASV	4.1.2.6	x				
TYPE	4.1.2.13	x				1
STRU	4.1.2.13	x				1
MODE	4.1.2.13	x				1
RETR	4.1.2.13	x				
STOR	4.1.2.13	x				
STOU	959 5.3.1			x		
APPE	4.1.2.13	x				
ALLO	959 5.3.1			x		
REST	959 5.3.1			x		
RNFR	4.1.2.13	x				
RNTO	4.1.2.13	x				
ABOR	959 5.3.1			x		
DELE	4.1.2.13	x				
RMD	4.1.2.13	x				
MKD	4.1.2.13	x				
PWD	4.1.2.13	x				
LIST	4.1.2.13	x				
NLST	4.1.2.13	x				
SITE	4.1.2.8			x		
STAT	4.1.2.13	x				
SYST	4.1.2.13	x				
HELP	4.1.2.13	x				
NOOP	4.1.2.13	x				

User Interface:					
Arbitrary pathnames	4.1.4.1	x			
Implement "QUOTE" command	4.1.4.2	x			
Transfer control commands immediately	4.1.4.2		x		
Display error messages to user	4.1.4.3		x		
Verbose mode	4.1.4.3		x		
Maintain synchronization with server	4.1.4.4		x		

Footnotes:

(1) For the values shown earlier.

(2) Here m is number of bits in a memory word.

(3) Required for host with record-structured file system, optional otherwise.

4.2 TRIVIAL FILE TRANSFER PROTOCOL -- TFTP

4.2.1 INTRODUCTION

The Trivial File Transfer Protocol TFTP is defined in RFC-783 [TFTP:1].

TFTP provides its own reliable delivery with UDP as its transport protocol, using a simple stop-and-wait acknowledgment system. Since TFTP has an effective window of only one 512 octet segment, it can provide good performance only over paths that have a small delay*bandwidth product. The TFTP file interface is very simple, providing no access control or security.

TFTP's most important application is bootstrapping a host over a local network, since it is simple and small enough to be easily implemented in EPROM [BOOT:1, BOOT:2]. Vendors are urged to support TFTP for booting.

4.2.2 PROTOCOL WALK-THROUGH

The TFTP specification [TFTP:1] is written in an open style, and does not fully specify many parts of the protocol.

4.2.2.1 Transfer Modes: RFC-783, Page 3

The transfer mode "mail" SHOULD NOT be supported.

4.2.2.2 UDP Header: RFC-783, Page 17

The Length field of a UDP header is incorrectly defined; it includes the UDP header length (8).

4.2.3 SPECIFIC ISSUES

4.2.3.1 Sorcerer's Apprentice Syndrome

There is a serious bug, known as the "Sorcerer's Apprentice Syndrome," in the protocol specification. While it does not cause incorrect operation of the transfer (the file will always be transferred correctly if the transfer completes), this bug may cause excessive retransmission, which may cause the transfer to time out.

Implementations MUST contain the fix for this problem: the sender (i.e., the side originating the DATA packets) must never resend the current DATA packet on receipt of a

duplicate ACK.

DISCUSSION:

The bug is caused by the protocol rule that either side, on receiving an old duplicate datagram, may resend the current datagram. If a packet is delayed in the network but later successfully delivered after either side has timed out and retransmitted a packet, a duplicate copy of the response may be generated. If the other side responds to this duplicate with a duplicate of its own, then every datagram will be sent in duplicate for the remainder of the transfer (unless a datagram is lost, breaking the repetition). Worse yet, since the delay is often caused by congestion, this duplicate transmission will usually causes more congestion, leading to more delayed packets, etc.

The following example may help to clarify this problem.

```
    TFTP A                  TFTP B

(1)  Receive ACK X-1
     Send DATA X
(2)                          Receive DATA X
                             Send ACK X
       (ACK X is delayed in network,
        and  A times out):
(3)  Retransmit DATA X

(4)                          Receive DATA X again
                             Send ACK X again
(5)  Receive (delayed) ACK X
     Send DATA X+1
(6)                          Receive DATA X+1
                             Send ACK X+1
(7)  Receive ACK X again
     Send DATA X+1 again
(8)                          Receive DATA X+1 again
                             Send ACK X+1 again
(9)  Receive ACK X+1
     Send DATA X+2
(10)                         Receive DATA X+2
                             Send ACK X+3
(11) Receive ACK X+1 again
     Send DATA X+2 again
(12)                         Receive DATA X+2 again
                             Send ACK X+3 again
```

Notice that once the delayed ACK arrives, the protocol settles down to duplicate all further packets (sequences 5-8 and 9-12). The problem is caused not by either side timing out, but by both sides retransmitting the current packet when they receive a duplicate.

The fix is to break the retransmission loop, as indicated above. This is analogous to the behavior of TCP. It is then possible to remove the retransmission timer on the receiver, since the resent ACK will never cause any action; this is a useful simplification where TFTP is used in a bootstrap program. It is OK to allow the timer to remain, and it may be helpful if the retransmitted ACK replaces one that was genuinely lost in the network. The sender still requires a retransmit timer, of course.

4.2.3.2 Timeout Algorithms

A TFTP implementation MUST use an adaptive timeout.

IMPLEMENTATION:
TCP retransmission algorithms provide a useful base to work from. At least an exponential backoff of retransmission timeout is necessary.

4.2.3.3 Extensions

A variety of non-standard extensions have been made to TFTP, including additional transfer modes and a secure operation mode (with passwords). None of these have been standardized.

4.2.3.4 Access Control

A server TFTP implementation SHOULD include some configurable access control over what pathnames are allowed in TFTP operations.

4.2.3.5 Broadcast Request

A TFTP request directed to a broadcast address SHOULD be silently ignored.

DISCUSSION:
Due to the weak access control capability of TFTP, directed broadcasts of TFTP requests to random networks

could create a significant security hole.

4.2.4 TFTP REQUIREMENTS SUMMARY

FEATURE	SECTION			SHOULD	MUST	SHOULD NOT	MUST NOT	Footnote
Fix Sorcerer's Apprentice Syndrome	4.2.3.1	x						
Transfer modes:								
netascii	RFC-783	x						
octet	RFC-783	x						
mail	4.2.2.1				x			
extensions	4.2.3.3			x				
Use adaptive timeout	4.2.3.2	x						
Configurable access control	4.2.3.4		x					
Silently ignore broadcast request	4.2.3.5		x					

5. ELECTRONIC MAIL -- SMTP and RFC-822

5.1 INTRODUCTION

In the TCP/IP protocol suite, electronic mail in a format specified in RFC-822 [SMTP:2] is transmitted using the Simple Mail Transfer Protocol (SMTP) defined in RFC-821 [SMTP:1].

While SMTP has remained unchanged over the years, the Internet community has made several changes in the way SMTP is used. In particular, the conversion to the Domain Name System (DNS) has caused changes in address formats and in mail routing. In this section, we assume familiarity with the concepts and terminology of the DNS, whose requirements are given in Section 6.1.

RFC-822 specifies the Internet standard format for electronic mail messages. RFC-822 supercedes an older standard, RFC-733, that may still be in use in a few places, although it is obsolete. The two formats are sometimes referred to simply by number ("822" and "733").

RFC-822 is used in some non-Internet mail environments with different mail transfer protocols than SMTP, and SMTP has also been adapted for use in some non-Internet environments. Note that this document presents the rules for the use of SMTP and RFC-822 for the Internet environment only; other mail environments that use these protocols may be expected to have their own rules.

5.2 PROTOCOL WALK-THROUGH

This section covers both RFC-821 and RFC-822.

The SMTP specification in RFC-821 is clear and contains numerous examples, so implementors should not find it difficult to understand. This section simply updates or annotates portions of RFC-821 to conform with current usage.

RFC-822 is a long and dense document, defining a rich syntax. Unfortunately, incomplete or defective implementations of RFC-822 are common. In fact, nearly all of the many formats of RFC-822 are actually used, so an implementation generally needs to recognize and correctly interpret all of the RFC-822 syntax.

5.2.1 The SMTP Model: RFC-821 Section 2

DISCUSSION:

Mail is sent by a series of request/response transactions between a client, the "sender-SMTP," and a server, the

"receiver-SMTP". These transactions pass (1) the message proper, which is composed of header and body, and (2) SMTP source and destination addresses, referred to as the "envelope".

The SMTP programs are analogous to Message Transfer Agents (MTAs) of X.400. There will be another level of protocol software, closer to the end user, that is responsible for composing and analyzing RFC-822 message headers; this component is known as the "User Agent" in X.400, and we use that term in this document. There is a clear logical distinction between the User Agent and the SMTP implementation, since they operate on different levels of protocol. Note, however, that this distinction is may not be exactly reflected the structure of typical implementations of Internet mail. Often there is a program known as the "mailer" that implements SMTP and also some of the User Agent functions; the rest of the User Agent functions are included in a user interface used for entering and reading mail.

The SMTP envelope is constructed at the originating site, typically by the User Agent when the message is first queued for the Sender-SMTP program. The envelope addresses may be derived from information in the message header, supplied by the user interface (e.g., to implement a bcc: request), or derived from local configuration information (e.g., expansion of a mailing list). The SMTP envelope cannot in general be re-derived from the header at a later stage in message delivery, so the envelope is transmitted separately from the message itself using the MAIL and RCPT commands of SMTP.

The text of RFC-821 suggests that mail is to be delivered to an individual user at a host. With the advent of the domain system and of mail routing using mail-exchange (MX) resource records, implementors should now think of delivering mail to a user at a domain, which may or may not be a particular host. This DOES NOT change the fact that SMTP is a host-to-host mail exchange protocol.

5.2.2 Canonicalization: RFC-821 Section 3.1

The domain names that a Sender-SMTP sends in MAIL and RCPT commands MUST have been "canonicalized," i.e., they must be fully-qualified principal names or domain literals, not nicknames or domain abbreviations. A canonicalized name either identifies a host directly or is an MX name; it cannot be a

CNAME.

5.2.3 VRFY and EXPN Commands: RFC-821 Section 3.3

A receiver-SMTP MUST implement VRFY and SHOULD implement EXPN (this requirement overrides RFC-821). However, there MAY be configuration information to disable VRFY and EXPN in a particular installation; this might even allow EXPN to be disabled for selected lists.

A new reply code is defined for the VRFY command:

252 Cannot VRFY user (e.g., info is not local), but will take message for this user and attempt delivery.

DISCUSSION:

SMTP users and administrators make regular use of these commands for diagnosing mail delivery problems. With the increasing use of multi-level mailing list expansion (sometimes more than two levels), EXPN has been increasingly important for diagnosing inadvertent mail loops. On the other hand, some feel that EXPN represents a significant privacy, and perhaps even a security, exposure.

5.2.4 SEND, SOML, and SAML Commands: RFC-821 Section 3.4

An SMTP MAY implement the commands to send a message to a user's terminal: SEND, SOML, and SAML.

DISCUSSION:

It has been suggested that the use of mail relaying through an MX record is inconsistent with the intent of SEND to deliver a message immediately and directly to a user's terminal. However, an SMTP receiver that is unable to write directly to the user terminal can return a "251 User Not Local" reply to the RCPT following a SEND, to inform the originator of possibly deferred delivery.

5.2.5 HELO Command: RFC-821 Section 3.5

The sender-SMTP MUST ensure that the <domain> parameter in a HELO command is a valid principal host domain name for the client host. As a result, the receiver-SMTP will not have to perform MX resolution on this name in order to validate the HELO parameter.

The HELO receiver MAY verify that the HELO parameter really

corresponds to the IP address of the sender. However, the receiver MUST NOT refuse to accept a message, even if the sender's HELO command fails verification.

DISCUSSION:

Verifying the HELO parameter requires a domain name lookup and may therefore take considerable time. An alternative tool for tracking bogus mail sources is suggested below (see "DATA Command").

Note also that the HELO argument is still required to have valid <domain> syntax, since it will appear in a Received: line; otherwise, a 501 error is to be sent.

IMPLEMENTATION:

When HELO parameter validation fails, a suggested procedure is to insert a note about the unknown authenticity of the sender into the message header (e.g., in the "Received:" line).

5.2.6 Mail Relay: RFC-821 Section 3.6

We distinguish three types of mail (store-and-) forwarding:

(1) A simple forwarder or "mail exchanger" forwards a message using private knowledge about the recipient; see section 3.2 of RFC-821.

(2) An SMTP mail "relay" forwards a message within an SMTP mail environment as the result of an explicit source route (as defined in section 3.6 of RFC-821). The SMTP relay function uses the "@...:" form of source route from RFC-822 (see Section 5.2.19 below).

(3) A mail "gateway" passes a message between different environments. The rules for mail gateways are discussed below in Section 5.3.7.

An Internet host that is forwarding a message but is not a gateway to a different mail environment (i.e., it falls under (1) or (2)) SHOULD NOT alter any existing header fields, although the host will add an appropriate Received: line as required in Section 5.2.8.

A Sender-SMTP SHOULD NOT send a RCPT TO: command containing an explicit source route using the "@...:" address form. Thus, the relay function defined in section 3.6 of RFC-821 should not be used.

DISCUSSION:

The intent is to discourage all source routing and to abolish explicit source routing for mail delivery within the Internet environment. Source-routing is unnecessary; the simple target address "user@domain" should always suffice. This is the result of an explicit architectural decision to use universal naming rather than source routing for mail. Thus, SMTP provides end-to-end connectivity, and the DNS provides globally-unique, location-independent names. MX records handle the major case where source routing might otherwise be needed.

A receiver-SMTP MUST accept the explicit source route syntax in the envelope, but it MAY implement the relay function as defined in section 3.6 of RFC-821. If it does not implement the relay function, it SHOULD attempt to deliver the message directly to the host to the right of the right-most "@" sign.

DISCUSSION:

For example, suppose a host that does not implement the relay function receives a message with the SMTP command: "RCPT TO:<@ALPHA,@BETA:joe@GAMMA>", where ALPHA, BETA, and GAMMA represent domain names. Rather than immediately refusing the message with a 550 error reply as suggested on page 20 of RFC-821, the host should try to forward the message to GAMMA directly, using: "RCPT TO:<joe@GAMMA>". Since this host does not support relaying, it is not required to update the reverse path.

Some have suggested that source routing may be needed occasionally for manually routing mail around failures; however, the reality and importance of this need is controversial. The use of explicit SMTP mail relaying for this purpose is discouraged, and in fact it may not be successful, as many host systems do not support it. Some have used the "%-hack" (see Section 5.2.16) for this purpose.

5.2.7 RCPT Command: RFC-821 Section 4.1.1

A host that supports a receiver-SMTP MUST support the reserved mailbox "Postmaster".

The receiver-SMTP MAY verify RCPT parameters as they arrive; however, RCPT responses MUST NOT be delayed beyond a reasonable time (see Section 5.3.2).

Therefore, a "250 OK" response to a RCPT does not necessarily

imply that the delivery address(es) are valid. Errors found after message acceptance will be reported by mailing a notification message to an appropriate address (see Section 5.3.3).

DISCUSSION:

The set of conditions under which a RCPT parameter can be validated immediately is an engineering design choice. Reporting destination mailbox errors to the Sender-SMTP before mail is transferred is generally desirable to save time and network bandwidth, but this advantage is lost if RCPT verification is lengthy.

For example, the receiver can verify immediately any simple local reference, such as a single locally-registered mailbox. On the other hand, the "reasonable time" limitation generally implies deferring verification of a mailing list until after the message has been transferred and accepted, since verifying a large mailing list can take a very long time. An implementation might or might not choose to defer validation of addresses that are non-local and therefore require a DNS lookup. If a DNS lookup is performed but a soft domain system error (e.g., timeout) occurs, validity must be assumed.

5.2.8 DATA Command: RFC-821 Section 4.1.1

Every receiver-SMTP (not just one that "accepts a message for relaying or for final delivery" [SMTP:1]) MUST insert a "Received:" line at the beginning of a message. In this line, called a "time stamp line" in RFC-821:

* The FROM field SHOULD contain both (1) the name of the source host as presented in the HELO command and (2) a domain literal containing the IP address of the source, determined from the TCP connection.

* The ID field MAY contain an "@" as suggested in RFC-822, but this is not required.

* The FOR field MAY contain a list of <path> entries when multiple RCPT commands have been given.

An Internet mail program MUST NOT change a Received: line that was previously added to the message header.

DISCUSSION:

Including both the source host and the IP source address in the Received: line may provide enough information for tracking illicit mail sources and eliminate a need to explicitly verify the HELO parameter.

Received: lines are primarily intended for humans tracing mail routes, primarily of diagnosis of faults. See also the discussion under 5.3.7.

When the receiver-SMTP makes "final delivery" of a message, then it MUST pass the MAIL FROM: address from the SMTP envelope with the message, for use if an error notification message must be sent later (see Section 5.3.3). There is an analogous requirement when gatewaying from the Internet into a different mail environment; see Section 5.3.7.

DISCUSSION:

Note that the final reply to the DATA command depends only upon the successful transfer and storage of the message. Any problem with the destination address(es) must either (1) have been reported in an SMTP error reply to the RCPT command(s), or (2) be reported in a later error message mailed to the originator.

IMPLEMENTATION:

The MAIL FROM: information may be passed as a parameter or in a Return-Path: line inserted at the beginning of the message.

5.2.9 Command Syntax: RFC-821 Section 4.1.2

The syntax shown in RFC-821 for the MAIL FROM: command omits the case of an empty path: "MAIL FROM: <>" (see RFC-821 Page 15). An empty reverse path MUST be supported.

5.2.10 SMTP Replies: RFC-821 Section 4.2

A receiver-SMTP SHOULD send only the reply codes listed in section 4.2.2 of RFC-821 or in this document. A receiver-SMTP SHOULD use the text shown in examples in RFC-821 whenever appropriate.

A sender-SMTP MUST determine its actions only by the reply code, not by the text (except for 251 and 551 replies); any text, including no text at all, must be acceptable. The space (blank) following the reply code is considered part of the text. Whenever possible, a sender-SMTP SHOULD test only the

first digit of the reply code, as specified in Appendix E of RFC-821.

DISCUSSION:
Interoperability problems have arisen with SMTP systems using reply codes that are not listed explicitly in RFC-821 Section 4.3 but are legal according to the theory of reply codes explained in Appendix E.

5.2.11 Transparency: RFC-821 Section 4.5.2

Implementors MUST be sure that their mail systems always add and delete periods to ensure message transparency.

5.2.12 WKS Use in MX Processing: RFC-974, p. 5

RFC-974 [SMTP:3] recommended that the domain system be queried for WKS ("Well-Known Service") records, to verify that each proposed mail target does support SMTP. Later experience has shown that WKS is not widely supported, so the WKS step in MX processing SHOULD NOT be used.

The following are notes on RFC-822, organized by section of that document.

5.2.13 RFC-822 Message Specification: RFC-822 Section 4

The syntax shown for the Return-path line omits the possibility of a null return path, which is used to prevent looping of error notifications (see Section 5.3.3). The complete syntax is:

```
return = "Return-path"  ":" route-addr
       / "Return-path"  ":" "<" ">"
```

The set of optional header fields is hereby expanded to include the Content-Type field defined in RFC-1049 [SMTP:7]. This field "allows mail reading systems to automatically identify the type of a structured message body and to process it for display accordingly". [SMTP:7] A User Agent MAY support this field.

5.2.14 RFC-822 Date and Time Specification: RFC-822 Section 5

The syntax for the date is hereby changed to:

```
date = 1*2DIGIT month 2*4DIGIT
```

All mail software SHOULD use 4-digit years in dates, to ease the transition to the next century.

There is a strong trend towards the use of numeric timezone indicators, and implementations SHOULD use numeric timezones instead of timezone names. However, all implementations MUST accept either notation. If timezone names are used, they MUST be exactly as defined in RFC-822.

The military time zones are specified incorrectly in RFC-822: they count the wrong way from UT (the signs are reversed). As a result, military time zones in RFC-822 headers carry no information.

Finally, note that there is a typo in the definition of "zone" in the syntax summary of appendix D; the correct definition occurs in Section 3 of RFC-822.

5.2.15 RFC-822 Syntax Change: RFC-822 Section 6.1

The syntactic definition of "mailbox" in RFC-822 is hereby changed to:

```
mailbox =  addr-spec            ; simple address
        / [phrase] route-addr   ; name & addr-spec
```

That is, the phrase preceding a route address is now OPTIONAL. This change makes the following header field legal, for example:

```
From: <craig@nnsc.nsf.net>
```

5.2.16 RFC-822 Local-part: RFC-822 Section 6.2

The basic mailbox address specification has the form: "local-part@domain". Here "local-part", sometimes called the "left-hand side" of the address, is domain-dependent.

A host that is forwarding the message but is not the destination host implied by the right-hand side "domain" MUST NOT interpret or modify the "local-part" of the address.

When mail is to be gatewayed from the Internet mail environment into a foreign mail environment (see Section 5.3.7), routing information for that foreign environment MAY be embedded within the "local-part" of the address. The gateway will then interpret this local part appropriately for the foreign mail environment.

DISCUSSION:

Although source routes are discouraged within the Internet (see Section 5.2.6), there are non-Internet mail environments whose delivery mechanisms do depend upon source routes. Source routes for extra-Internet environments can generally be buried in the "local-part" of the address (see Section 5.2.16) while mail traverses the Internet. When the mail reaches the appropriate Internet mail gateway, the gateway will interpret the local-part and build the necessary address or route for the target mail environment.

For example, an Internet host might send mail to: "a!b!c!user@gateway-domain". The complex local part "a!b!c!user" would be uninterpreted within the Internet domain, but could be parsed and understood by the specified mail gateway.

An embedded source route is sometimes encoded in the "local-part" using "%" as a right-binding routing operator. For example, in:

```
user%domain%relay3%relay2@relay1
```

the "%" convention implies that the mail is to be routed from "relay1" through "relay2", "relay3", and finally to "user" at "domain". This is commonly known as the "%-hack". It is suggested that "%" have lower precedence than any other routing operator (e.g., "!") hidden in the local-part; for example, "a!b%c" would be interpreted as "(a!b)%c".

Only the target host (in this case, "relay1") is permitted to analyze the local-part "user%domain%relay3%relay2".

5.2.17 Domain Literals: RFC-822 Section 6.2.3

A mailer MUST be able to accept and parse an Internet domain literal whose content ("dtext"; see RFC-822) is a dotted-decimal host address. This satisfies the requirement of Section 2.1 for the case of mail.

An SMTP MUST accept and recognize a domain literal for any of its own IP addresses.

5.2.18 Common Address Formatting Errors: RFC-822 Section 6.1

Errors in formatting or parsing 822 addresses are unfortunately common. This section mentions only the most common errors. A User Agent MUST accept all valid RFC-822 address formats, and MUST NOT generate illegal address syntax.

- A common error is to leave out the semicolon after a group identifier.

- Some systems fail to fully-qualify domain names in messages they generate. The right-hand side of an "@" sign in a header address field MUST be a fully-qualified domain name.

  For example, some systems fail to fully-qualify the From: address; this prevents a "reply" command in the user interface from automatically constructing a return address.

  DISCUSSION:
  Although RFC-822 allows the local use of abbreviated domain names within a domain, the application of RFC-822 in Internet mail does not allow this. The intent is that an Internet host must not send an SMTP message header containing an abbreviated domain name in an address field. This allows the address fields of the header to be passed without alteration across the Internet, as required in Section 5.2.6.

- Some systems mis-parse multiple-hop explicit source routes such as:

  ```
  @relay1,@relay2,@relay3:user@domain.
  ```

- Some systems over-qualify domain names by adding a trailing dot to some or all domain names in addresses or message-ids. This violates RFC-822 syntax.

5.2.19 Explicit Source Routes: RFC-822 Section 6.2.7

Internet host software SHOULD NOT create an RFC-822 header containing an address with an explicit source route, but MUST accept such headers for compatibility with earlier systems.

DISCUSSION:

In an understatement, RFC-822 says "The use of explicit source routing is discouraged". Many hosts implemented RFC-822 source routes incorrectly, so the syntax cannot be used unambiguously in practice. Many users feel the syntax is ugly. Explicit source routes are not needed in the mail envelope for delivery; see Section 5.2.6. For all these reasons, explicit source routes using the RFC-822 notations are not to be used in Internet mail headers.

As stated in Section 5.2.16, it is necessary to allow an explicit source route to be buried in the local-part of an address, e.g., using the "%-hack", in order to allow mail to be gatewayed into another environment in which explicit source routing is necessary. The vigilant will observe that there is no way for a User Agent to detect and prevent the use of such implicit source routing when the destination is within the Internet. We can only discourage source routing of any kind within the Internet, as unnecessary and undesirable.

5.3 SPECIFIC ISSUES

5.3.1 SMTP Queueing Strategies

The common structure of a host SMTP implementation includes user mailboxes, one or more areas for queueing messages in transit, and one or more daemon processes for sending and receiving mail. The exact structure will vary depending on the needs of the users on the host and the number and size of mailing lists supported by the host. We describe several optimizations that have proved helpful, particularly for mailers supporting high traffic levels.

Any queueing strategy MUST include:

- Timeouts on all activities. See Section 5.3.2.
- Never sending error messages in response to error messages.

5.3.1.1 Sending Strategy

The general model of a sender-SMTP is one or more processes that periodically attempt to transmit outgoing mail. In a typical system, the program that composes a message has some method for requesting immediate attention for a new piece of outgoing mail, while mail that cannot be transmitted

immediately MUST be queued and periodically retried by the sender. A mail queue entry will include not only the message itself but also the envelope information.

The sender MUST delay retrying a particular destination after one attempt has failed. In general, the retry interval SHOULD be at least 30 minutes; however, more sophisticated and variable strategies will be beneficial when the sender-SMTP can determine the reason for non-delivery.

Retries continue until the message is transmitted or the sender gives up; the give-up time generally needs to be at least 4-5 days. The parameters to the retry algorithm MUST be configurable.

A sender SHOULD keep a list of hosts it cannot reach and corresponding timeouts, rather than just retrying queued mail items.

DISCUSSION:

Experience suggests that failures are typically transient (the target system has crashed), favoring a policy of two connection attempts in the first hour the message is in the queue, and then backing off to once every two or three hours.

The sender-SMTP can shorten the queueing delay by cooperation with the receiver-SMTP. In particular, if mail is received from a particular address, it is good evidence that any mail queued for that host can now be sent.

The strategy may be further modified as a result of multiple addresses per host (see Section 5.3.4), to optimize delivery time vs. resource usage.

A sender-SMTP may have a large queue of messages for each unavailable destination host, and if it retried all these messages in every retry cycle, there would be excessive Internet overhead and the daemon would be blocked for a long period. Note that an SMTP can generally determine that a delivery attempt has failed only after a timeout of a minute or more; a one minute timeout per connection will result in a very large delay if it is repeated for dozens or even hundreds of queued messages.

When the same message is to be delivered to several users on the same host, only one copy of the message SHOULD be transmitted. That is, the sender-SMTP should use the command sequence: RCPT, RCPT,... RCPT, DATA instead of the sequence: RCPT, DATA, RCPT, DATA,... RCPT, DATA. Implementation of this efficiency feature is strongly urged.

Similarly, the sender-SMTP MAY support multiple concurrent outgoing mail transactions to achieve timely delivery. However, some limit SHOULD be imposed to protect the host from devoting all its resources to mail.

The use of the different addresses of a multihomed host is discussed below.

5.3.1.2 Receiving strategy

The receiver-SMTP SHOULD attempt to keep a pending listen on the SMTP port at all times. This will require the support of multiple incoming TCP connections for SMTP. Some limit MAY be imposed.

IMPLEMENTATION:

When the receiver-SMTP receives mail from a particular host address, it could notify the sender-SMTP to retry any mail pending for that host address.

5.3.2 Timeouts in SMTP

There are two approaches to timeouts in the sender-SMTP: (a) limit the time for each SMTP command separately, or (b) limit the time for the entire SMTP dialogue for a single mail message. A sender-SMTP SHOULD use option (a), per-command timeouts. Timeouts SHOULD be easily reconfigurable, preferably without recompiling the SMTP code.

DISCUSSION:

Timeouts are an essential feature of an SMTP implementation. If the timeouts are too long (or worse, there are no timeouts), Internet communication failures or software bugs in receiver-SMTP programs can tie up SMTP processes indefinitely. If the timeouts are too short, resources will be wasted with attempts that time out part way through message delivery.

If option (b) is used, the timeout has to be very large, e.g., an hour, to allow time to expand very large mailing lists. The timeout may also need to increase linearly

with the size of the message, to account for the time to transmit a very large message. A large fixed timeout leads to two problems: a failure can still tie up the sender for a very long time, and very large messages may still spuriously time out (which is a wasteful failure!).

Using the recommended option (a), a timer is set for each SMTP command and for each buffer of the data transfer. The latter means that the overall timeout is inherently proportional to the size of the message.

Based on extensive experience with busy mail-relay hosts, the minimum per-command timeout values SHOULD be as follows:

- Initial 220 Message: 5 minutes

  A Sender-SMTP process needs to distinguish between a failed TCP connection and a delay in receiving the initial 220 greeting message. Many receiver-SMTPs will accept a TCP connection but delay delivery of the 220 message until their system load will permit more mail to be processed.

- MAIL Command: 5 minutes

- RCPT Command: 5 minutes

  A longer timeout would be required if processing of mailing lists and aliases were not deferred until after the message was accepted.

- DATA Initiation: 2 minutes

  This is while awaiting the "354 Start Input" reply to a DATA command.

- Data Block: 3 minutes

  This is while awaiting the completion of each TCP SEND call transmitting a chunk of data.

- DATA Termination: 10 minutes.

  This is while awaiting the "250 OK" reply. When the receiver gets the final period terminating the message data, it typically performs processing to deliver the message to a user mailbox. A spurious timeout at this point would be very wasteful, since the message has been

successfully sent.

A receiver-SMTP SHOULD have a timeout of at least 5 minutes while it is awaiting the next command from the sender.

5.3.3 Reliable Mail Receipt

When the receiver-SMTP accepts a piece of mail (by sending a "250 OK" message in response to DATA), it is accepting responsibility for delivering or relaying the message. It must take this responsibility seriously, i.e., it MUST NOT lose the message for frivolous reasons, e.g., because the host later crashes or because of a predictable resource shortage.

If there is a delivery failure after acceptance of a message, the receiver-SMTP MUST formulate and mail a notification message. This notification MUST be sent using a null ("<>") reverse path in the envelope; see Section 3.6 of RFC-821. The recipient of this notification SHOULD be the address from the envelope return path (or the Return-Path: line). However, if this address is null ("<>"), the receiver-SMTP MUST NOT send a notification. If the address is an explicit source route, it SHOULD be stripped down to its final hop.

DISCUSSION:

For example, suppose that an error notification must be sent for a message that arrived with: "MAIL FROM:<@a,@b:user@d>". The notification message should be sent to: "RCPT TO:<user@d>".

Some delivery failures after the message is accepted by SMTP will be unavoidable. For example, it may be impossible for the receiver-SMTP to validate all the delivery addresses in RCPT command(s) due to a "soft" domain system error or because the target is a mailing list (see earlier discussion of RCPT).

To avoid receiving duplicate messages as the result of timeouts, a receiver-SMTP MUST seek to minimize the time required to respond to the final "." that ends a message transfer. See RFC-1047 [SMTP:4] for a discussion of this problem.

5.3.4 Reliable Mail Transmission

To transmit a message, a sender-SMTP determines the IP address of the target host from the destination address in the envelope. Specifically, it maps the string to the right of the

"@" sign into an IP address. This mapping or the transfer itself may fail with a soft error, in which case the sender-SMTP will requeue the outgoing mail for a later retry, as required in Section 5.3.1.1.

When it succeeds, the mapping can result in a list of alternative delivery addresses rather than a single address, because of (a) multiple MX records, (b) multihoming, or both. To provide reliable mail transmission, the sender-SMTP MUST be able to try (and retry) each of the addresses in this list in order, until a delivery attempt succeeds. However, there MAY also be a configurable limit on the number of alternate addresses that can be tried. In any case, a host SHOULD try at least two addresses.

The following information is to be used to rank the host addresses:

(1) Multiple MX Records -- these contain a preference indication that should be used in sorting. If there are multiple destinations with the same preference and there is no clear reason to favor one (e.g., by address preference), then the sender-SMTP SHOULD pick one at random to spread the load across multiple mail exchanges for a specific organization; note that this is a refinement of the procedure in [DNS:3].

(2) Multihomed host -- The destination host (perhaps taken from the preferred MX record) may be multihomed, in which case the domain name resolver will return a list of alternative IP addresses. It is the responsibility of the domain name resolver interface (see Section 6.1.3.4 below) to have ordered this list by decreasing preference, and SMTP MUST try them in the order presented.

DISCUSSION:

Although the capability to try multiple alternative addresses is required, there may be circumstances where specific installations want to limit or disable the use of alternative addresses. The question of whether a sender should attempt retries using the different addresses of a multihomed host has been controversial. The main argument for using the multiple addresses is that it maximizes the probability of timely delivery, and indeed sometimes the probability of any delivery; the counter argument is that it may result in unnecessary resource use.

Note that resource use is also strongly determined by the

sending strategy discussed in Section 5.3.1.

5.3.5 Domain Name Support

SMTP implementations MUST use the mechanism defined in Section 6.1 for mapping between domain names and IP addresses. This means that every Internet SMTP MUST include support for the Internet DNS.

In particular, a sender-SMTP MUST support the MX record scheme [SMTP:3]. See also Section 7.4 of [DNS:2] for information on domain name support for SMTP.

5.3.6 Mailing Lists and Aliases

An SMTP-capable host SHOULD support both the alias and the list form of address expansion for multiple delivery. When a message is delivered or forwarded to each address of an expanded list form, the return address in the envelope ("MAIL FROM:") MUST be changed to be the address of a person who administers the list, but the message header MUST be left unchanged; in particular, the "From" field of the message is unaffected.

DISCUSSION:

An important mail facility is a mechanism for multi-destination delivery of a single message, by transforming or "expanding" a pseudo-mailbox address into a list of destination mailbox addresses. When a message is sent to such a pseudo-mailbox (sometimes called an "exploder"), copies are forwarded or redistributed to each mailbox in the expanded list. We classify such a pseudo-mailbox as an "alias" or a "list", depending upon the expansion rules:

(a) Alias

To expand an alias, the recipient mailer simply replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn; the rest of the envelope and the message body are left unchanged. The message is then delivered or forwarded to each expanded address.

(b) List

A mailing list may be said to operate by "redistribution" rather than by "forwarding". To

expand a list, the recipient mailer replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn. The return address in the envelope is changed so that all error messages generated by the final deliveries will be returned to a list administrator, not to the message originator, who generally has no control over the contents of the list and will typically find error messages annoying.

5.3.7 Mail Gatewaying

Gatewaying mail between different mail environments, i.e., different mail formats and protocols, is complex and does not easily yield to standardization. See for example [SMTP:5a], [SMTP:5b]. However, some general requirements may be given for a gateway between the Internet and another mail environment.

(A) Header fields MAY be rewritten when necessary as messages are gatewayed across mail environment boundaries.

DISCUSSION:

This may involve interpreting the local-part of the destination address, as suggested in Section 5.2.16.

The other mail systems gatewayed to the Internet generally use a subset of RFC-822 headers, but some of them do not have an equivalent to the SMTP envelope. Therefore, when a message leaves the Internet environment, it may be necessary to fold the SMTP envelope information into the message header. A possible solution would be to create new header fields to carry the envelope information (e.g., "X-SMTP-MAIL:" and "X-SMTP-RCPT:"); however, this would require changes in mail programs in the foreign environment.

(B) When forwarding a message into or out of the Internet environment, a gateway MUST prepend a Received: line, but it MUST NOT alter in any way a Received: line that is already in the header.

DISCUSSION:

This requirement is a subset of the general "Received:" line requirement of Section 5.2.8; it is restated here for emphasis.

Received: fields of messages originating from other

environments may not conform exactly to RFC822. However, the most important use of Received: lines is for debugging mail faults, and this debugging can be severely hampered by well-meaning gateways that try to "fix" a Received: line.

The gateway is strongly encouraged to indicate the environment and protocol in the "via" clauses of Received field(s) that it supplies.

(C) From the Internet side, the gateway SHOULD accept all valid address formats in SMTP commands and in RFC-822 headers, and all valid RFC-822 messages. Although a gateway must accept an RFC-822 explicit source route ("@...:" format) in either the RFC-822 header or in the envelope, it MAY or may not act on the source route; see Sections 5.2.6 and 5.2.19.

DISCUSSION:
It is often tempting to restrict the range of addresses accepted at the mail gateway to simplify the translation into addresses for the remote environment. This practice is based on the assumption that mail users have control over the addresses their mailers send to the mail gateway. In practice, however, users have little control over the addresses that are finally sent; their mailers are free to change addresses into any legal RFC-822 format.

(D) The gateway MUST ensure that all header fields of a message that it forwards into the Internet meet the requirements for Internet mail. In particular, all addresses in "From:", "To:", "Cc:", etc., fields must be transformed (if necessary) to satisfy RFC-822 syntax, and they must be effective and useful for sending replies.

(E) The translation algorithm used to convert mail from the Internet protocols to another environment's protocol SHOULD try to ensure that error messages from the foreign mail environment are delivered to the return path from the SMTP envelope, not to the sender listed in the "From:" field of the RFC-822 message.

DISCUSSION:
Internet mail lists usually place the address of the mail list maintainer in the envelope but leave the

original message header intact (with the "From:" field containing the original sender). This yields the behavior the average recipient expects: a reply to the header gets sent to the original sender, not to a mail list maintainer; however, errors get sent to the maintainer (who can fix the problem) and not the sender (who probably cannot).

(F) Similarly, when forwarding a message from another environment into the Internet, the gateway SHOULD set the envelope return path in accordance with an error message return address, if any, supplied by the foreign environment.

5.3.8 Maximum Message Size

Mailer software MUST be able to send and receive messages of at least 64K bytes in length (including header), and a much larger maximum size is highly desirable.

DISCUSSION:

Although SMTP does not define the maximum size of a message, many systems impose implementation limits.

The current de facto minimum limit in the Internet is 64K bytes. However, electronic mail is used for a variety of purposes that create much larger messages. For example, mail is often used instead of FTP for transmitting ASCII files, and in particular to transmit entire documents. As a result, messages can be 1 megabyte or even larger. We note that the present document together with its lower-layer companion contains 0.5 megabytes.

5.4 SMTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHLD	MAY	SHLD NOT	MUST NOT	Footnote
RECEIVER-SMTP:							
Implement VRFY	5.2.3	x					
Implement EXPN	5.2.3		x				
EXPN, VRFY configurable	5.2.3			x			
Implement SEND, SOML, SAML	5.2.4			x			
Verify HELO parameter	5.2.5			x			
Refuse message with bad HELO	5.2.5					x	
Accept explicit src-route syntax in env.	5.2.6	x					
Support "postmaster"	5.2.7	x					
Process RCPT when received (except lists)	5.2.7			x			
Long delay of RCPT responses	5.2.7					x	
Add Received: line	5.2.8	x					
Received: line include domain literal	5.2.8		x				
Change previous Received: line	5.2.8					x	
Pass Return-Path info (final deliv/gwy)	5.2.8	x					
Support empty reverse path	5.2.9	x					
Send only official reply codes	5.2.10		x				
Send text from RFC-821 when appropriate	5.2.10		x				
Delete "." for transparency	5.2.11	x					
Accept and recognize self domain literal(s)	5.2.17	x					
Error message about error message	5.3.1					x	
Keep pending listen on SMTP port	5.3.1.2		x				
Provide limit on recv concurrency	5.3.1.2			x			
Wait at least 5 mins for next sender cmd	5.3.2		x				
Avoidable delivery failure after "250 OK"	5.3.3					x	
Send error notification msg after accept	5.3.3	x					
Send using null return path	5.3.3	x					
Send to envelope return path	5.3.3		x				
Send to null address	5.3.3					x	
Strip off explicit src route	5.3.3		x				
Minimize acceptance delay (RFC-1047)	5.3.3	x					

SENDER-SMTP:						
Canonicalized domain names in MAIL, RCPT	5.2.2	x				
Implement SEND, SOML, SAML	5.2.4			x		
Send valid principal host name in HELO	5.2.5	x				
Send explicit source route in RCPT TO:	5.2.6				x	
Use only reply code to determine action	5.2.10	x				
Use only high digit of reply code when poss.	5.2.10		x			
Add "." for transparency	5.2.11	x				
Retry messages after soft failure	5.3.1.1	x				
Delay before retry	5.3.1.1	x				
Configurable retry parameters	5.3.1.1	x				
Retry once per each queued dest host	5.3.1.1		x			
Multiple RCPT's for same DATA	5.3.1.1		x			
Support multiple concurrent transactions	5.3.1.1			x		
Provide limit on concurrency	5.3.1.1		x			
Timeouts on all activities	5.3.1	x				
Per-command timeouts	5.3.2		x			
Timeouts easily reconfigurable	5.3.2		x			
Recommended times	5.3.2		x			
Try alternate addr's in order	5.3.4	x				
Configurable limit on alternate tries	5.3.4			x		
Try at least two alternates	5.3.4		x			
Load-split across equal MX alternates	5.3.4		x			
Use the Domain Name System	5.3.5	x				
Support MX records	5.3.5	x				
Use WKS records in MX processing	5.2.12				x	
MAIL FORWARDING:						
Alter existing header field(s)	5.2.6				x	
Implement relay function: 821/section 3.6	5.2.6			x		
If not, deliver to RHS domain	5.2.6		x			
Interpret 'local-part' of addr	5.2.16					x
MAILING LISTS AND ALIASES						
Support both	5.3.6		x			
Report mail list error to local admin.	5.3.6	x				
MAIL GATEWAYS:						
Embed foreign mail route in local-part	5.2.16			x		
Rewrite header fields when necessary	5.3.7			x		
Prepend Received: line	5.3.7	x				
Change existing Received: line	5.3.7					x
Accept full RFC-822 on Internet side	5.3.7		x			
Act on RFC-822 explicit source route	5.3.7			x		

Send only valid RFC-822 on Internet side	5.3.7	x				
Deliver error msgs to envelope addr	5.3.7		x			
Set env return path from err return addr	5.3.7		x			
USER AGENT -- RFC-822						
Allow user to enter <route> address	5.2.6				x	
Support RFC-1049 Content Type field	5.2.13			x		
Use 4-digit years	5.2.14		x			
Generate numeric timezones	5.2.14		x			
Accept all timezones	5.2.14	x				
Use non-num timezones from RFC-822	5.2.14	x				
Omit phrase before route-addr	5.2.15			x		
Accept and parse dot.dec. domain literals	5.2.17	x				
Accept all RFC-822 address formats	5.2.18	x				
Generate invalid RFC-822 address format	5.2.18					x
Fully-qualified domain names in header	5.2.18	x				
Create explicit src route in header	5.2.19				x	
Accept explicit src route in header	5.2.19	x				
Send/recv at least 64KB messages	5.3.8	x				

6. SUPPORT SERVICES

6.1 DOMAIN NAME TRANSLATION

6.1.1 INTRODUCTION

Every host MUST implement a resolver for the Domain Name System (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa [DNS:1, DNS:2].

In addition to the DNS, a host MAY also implement a host name translation mechanism that searches a local Internet host table. See Section 6.1.3.8 for more information on this option.

DISCUSSION:

Internet host name translation was originally performed by searching local copies of a table of all hosts. This table became too large to update and distribute in a timely manner and too large to fit into many hosts, so the DNS was invented.

The DNS creates a distributed database used primarily for the translation between host names and host addresses. Implementation of DNS software is required. The DNS consists of two logically distinct parts: name servers and resolvers (although implementations often combine these two logical parts in the interest of efficiency) [DNS:2].

Domain name servers store authoritative data about certain sections of the database and answer queries about the data. Domain resolvers query domain name servers for data on behalf of user processes. Every host therefore needs a DNS resolver; some host machines will also need to run domain name servers. Since no name server has complete information, in general it is necessary to obtain information from more than one name server to resolve a query.

6.1.2 PROTOCOL WALK-THROUGH

An implementor must study references [DNS:1] and [DNS:2] carefully. They provide a thorough description of the theory, protocol, and implementation of the domain name system, and reflect several years of experience.

6.1.2.1 Resource Records with Zero TTL: RFC-1035 Section 3.2.1

All DNS name servers and resolvers MUST properly handle RRs with a zero TTL: return the RR to the client but do not cache it.

DISCUSSION:

Zero TTL values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached; they are useful for extremely volatile data.

6.1.2.2 QCLASS Values: RFC-1035 Section 3.2.5

A query with "QCLASS=*" SHOULD NOT be used unless the requestor is seeking data from more than one class. In particular, if the requestor is only interested in Internet data types, QCLASS=IN MUST be used.

6.1.2.3 Unused Fields: RFC-1035 Section 4.1.1

Unused fields in a query or response message MUST be zero.

6.1.2.4 Compression: RFC-1035 Section 4.1.4

Name servers MUST use compression in responses.

DISCUSSION:

Compression is essential to avoid overflowing UDP datagrams; see Section 6.1.3.2.

6.1.2.5 Misusing Configuration Info: RFC-1035 Section 6.1.2

Recursive name servers and full-service resolvers generally have some configuration information containing hints about the location of root or local name servers. An implementation MUST NOT include any of these hints in a response.

DISCUSSION:

Many implementors have found it convenient to store these hints as if they were cached data, but some neglected to ensure that this "cached data" was not included in responses. This has caused serious problems in the Internet when the hints were obsolete or incorrect.

6.1.3 SPECIFIC ISSUES

6.1.3.1 Resolver Implementation

A name resolver SHOULD be able to multiplex concurrent requests if the host supports concurrent processes.

In implementing a DNS resolver, one of two different models MAY optionally be chosen: a full-service resolver, or a stub resolver.

(A) Full-Service Resolver

A full-service resolver is a complete implementation of the resolver service, and is capable of dealing with communication failures, failure of individual name servers, location of the proper name server for a given name, etc. It must satisfy the following requirements:

- The resolver MUST implement a local caching function to avoid repeated remote access for identical requests, and MUST time out information in the cache.

- The resolver SHOULD be configurable with start-up information pointing to multiple root name servers and multiple name servers for the local domain. This insures that the resolver will be able to access the whole name space in normal cases, and will be able to access local domain information should the local network become disconnected from the rest of the Internet.

(B) Stub Resolver

A "stub resolver" relies on the services of a recursive name server on the connected network or a "nearby" network. This scheme allows the host to pass on the burden of the resolver function to a name server on another host. This model is often essential for less capable hosts, such as PCs, and is also recommended when the host is one of several workstations on a local network, because it allows all of the workstations to share the cache of the recursive name server and hence reduce the number of domain requests exported by the local network.

At a minimum, the stub resolver MUST be capable of directing its requests to redundant recursive name servers. Note that recursive name servers are allowed to restrict the sources of requests that they will honor, so the host administrator must verify that the service will be provided. Stub resolvers MAY implement caching if they choose, but if so, MUST timeout cached information.

6.1.3.2 Transport Protocols

DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries. Specifically, a DNS resolver or server that is sending a non-zone-transfer query MUST send a UDP query first. If the Answer section of the response is truncated and if the requester supports TCP, it SHOULD try the query again using TCP.

DNS servers MUST be able to service UDP queries and SHOULD be able to service TCP queries. A name server MAY limit the resources it devotes to TCP queries, but it SHOULD NOT refuse to service a TCP query just because it would have succeeded with UDP.

Truncated responses MUST NOT be saved (cached) and later used in such a way that the fact that they are truncated is lost.

DISCUSSION:

UDP is preferred over TCP for queries because UDP queries have much lower overhead, both in packet count and in connection state. The use of UDP is essential for heavily-loaded servers, especially the root servers. UDP also offers additional robustness, since a resolver can attempt several UDP queries to different servers for the cost of a single TCP query.

It is possible for a DNS response to be truncated, although this is a very rare occurrence in the present Internet DNS. Practically speaking, truncation cannot be predicted, since it is data-dependent. The dependencies include the number of RRs in the answer, the size of each RR, and the savings in space realized by the name compression algorithm. As a rule of thumb, truncation in NS and MX lists should not occur for answers containing 15 or fewer RRs.

Whether it is possible to use a truncated answer depends on the application. A mailer must not use a truncated MX response, since this could lead to mail loops.

Responsible practices can make UDP suffice in the vast majority of cases. Name servers must use compression in responses. Resolvers must differentiate truncation of the Additional section of a response (which only loses extra information) from truncation of the Answer section (which for MX records renders the response unusable by mailers). Database administrators should list only a reasonable number of primary names in lists of name servers, MX alternatives, etc.

However, it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

By private agreement, name servers and resolvers MAY arrange to use TCP for all traffic between themselves. TCP MUST be used for zone transfers.

A DNS server MUST have sufficient internal concurrency that it can continue to process UDP queries while awaiting a response or performing a zone transfer on an open TCP connection [DNS:2].

A server MAY support a UDP query that is delivered using an IP broadcast or multicast address. However, the Recursion Desired bit MUST NOT be set in a query that is multicast, and MUST be ignored by name servers receiving queries via a broadcast or multicast address. A host that sends broadcast or multicast DNS queries SHOULD send them only as occasional probes, caching the IP address(es) it obtains from the response(s) so it can normally send unicast queries.

DISCUSSION:

Broadcast or (especially) IP multicast can provide a way to locate nearby name servers without knowing their IP addresses in advance. However, general broadcasting of recursive queries can result in excessive and unnecessary load on both network and servers.

6.1.3.3 Efficient Resource Usage

The following requirements on servers and resolvers are very important to the health of the Internet as a whole, particularly when DNS services are invoked repeatedly by higher level automatic servers, such as mailers.

(1) The resolver MUST implement retransmission controls to insure that it does not waste communication bandwidth, and MUST impose finite bounds on the resources consumed to respond to a single request. See [DNS:2] pages 43-44 for specific recommendations.

(2) After a query has been retransmitted several times without a response, an implementation MUST give up and return a soft error to the application.

(3) All DNS name servers and resolvers SHOULD cache temporary failures, with a timeout period of the order of minutes.

DISCUSSION:
This will prevent applications that immediately retry soft failures (in violation of Section 2.2 of this document) from generating excessive DNS traffic.

(4) All DNS name servers and resolvers SHOULD cache negative responses that indicate the specified name, or data of the specified type, does not exist, as described in [DNS:2].

(5) When a DNS server or resolver retries a UDP query, the retry interval SHOULD be constrained by an exponential backoff algorithm, and SHOULD also have upper and lower bounds.

IMPLEMENTATION:
A measured RTT and variance (if available) should be used to calculate an initial retransmission interval. If this information is not available, a default of no less than 5 seconds should be used. Implementations may limit the retransmission interval, but this limit must exceed twice the Internet maximum segment lifetime plus service delay at the name server.

(6) When a resolver or server receives a Source Quench for

a query it has issued, it SHOULD take steps to reduce the rate of querying that server in the near future. A server MAY ignore a Source Quench that it receives as the result of sending a response datagram.

IMPLEMENTATION:
One recommended action to reduce the rate is to send the next query attempt to an alternate server, if there is one available. Another is to backoff the retry interval for the same server.

6.1.3.4 Multihomed Hosts

When the host name-to-address function encounters a host with multiple addresses, it SHOULD rank or sort the addresses using knowledge of the immediately connected network number(s) and any other applicable performance or history information.

DISCUSSION:
The different addresses of a multihomed host generally imply different Internet paths, and some paths may be preferable to others in performance, reliability, or administrative restrictions. There is no general way for the domain system to determine the best path. A recommended approach is to base this decision on local configuration information set by the system administrator.

IMPLEMENTATION:
The following scheme has been used successfully:

(a) Incorporate into the host configuration data a Network-Preference List, that is simply a list of networks in preferred order. This list may be empty if there is no preference.

(b) When a host name is mapped into a list of IP addresses, these addresses should be sorted by network number, into the same order as the corresponding networks in the Network-Preference List. IP addresses whose networks do not appear in the Network-Preference List should be placed at the end of the list.

6.1.3.5 Extensibility

DNS software MUST support all well-known, class-independent formats [DNS:2], and SHOULD be written to minimize the trauma associated with the introduction of new well-known types and local experimentation with non-standard types.

DISCUSSION:

The data types and classes used by the DNS are extensible, and thus new types will be added and old types deleted or redefined. Introduction of new data types ought to be dependent only upon the rules for compression of domain names inside DNS messages, and the translation between printable (i.e., master file) and internal formats for Resource Records (RRs).

Compression relies on knowledge of the format of data inside a particular RR. Hence compression must only be used for the contents of well-known, class-independent RRs, and must never be used for class-specific RRs or RR types that are not well-known. The owner name of an RR is always eligible for compression.

A name server may acquire, via zone transfer, RRs that the server doesn't know how to convert to printable format. A resolver can receive similar information as the result of queries. For proper operation, this data must be preserved, and hence the implication is that DNS software cannot use textual formats for internal storage.

The DNS defines domain name syntax very generally -- a string of labels each containing up to 63 8-bit octets, separated by dots, and with a maximum total of 255 octets. Particular applications of the DNS are permitted to further constrain the syntax of the domain names they use, although the DNS deployment has led to some applications allowing more general names. In particular, Section 2.1 of this document liberalizes slightly the syntax of a legal Internet host name that was defined in RFC-952 [DNS:4].

6.1.3.6 Status of RR Types

Name servers MUST be able to load all RR types except MD and MF from configuration files. The MD and MF types are obsolete and MUST NOT be implemented; in particular, name servers MUST NOT load these types from configuration files.

DISCUSSION:

The RR types MB, MG, MR, NULL, MINFO and RP are considered experimental, and applications that use the DNS cannot expect these RR types to be supported by most domains. Furthermore these types are subject to redefinition.

The TXT and WKS RR types have not been widely used by Internet sites; as a result, an application cannot rely on the the existence of a TXT or WKS RR in most domains.

6.1.3.7 Robustness

DNS software may need to operate in environments where the root servers or other servers are unavailable due to network connectivity or other problems. In this situation, DNS name servers and resolvers MUST continue to provide service for the reachable part of the name space, while giving temporary failures for the rest.

DISCUSSION:

Although the DNS is meant to be used primarily in the connected Internet, it should be possible to use the system in networks which are unconnected to the Internet. Hence implementations must not depend on access to root servers before providing service for local names.

6.1.3.8 Local Host Table

DISCUSSION:

A host may use a local host table as a backup or supplement to the DNS. This raises the question of which takes precedence, the DNS or the host table; the most flexible approach would make this a configuration option.

Typically, the contents of such a supplementary host table will be determined locally by the site. However, a publically-available table of Internet hosts is maintained by the DDN Network Information Center (DDN NIC), with a format documented in [DNS:4]. This table can be retrieved from the DDN NIC using a protocol described in [DNS:5]. It must be noted that this table contains only a small fraction of all Internet hosts. Hosts using this protocol to retrieve the DDN NIC host table should use the VERSION command to check if the

table has changed before requesting the entire table with the ALL command. The VERSION identifier should be treated as an arbitrary string and tested only for equality; no numerical sequence may be assumed.

The DDN NIC host table includes administrative information that is not needed for host operation and is therefore not currently included in the DNS database; examples include network and gateway entries. However, much of this additional information will be added to the DNS in the future. Conversely, the DNS provides essential services (in particular, MX records) that are not available from the DDN NIC host table.

6.1.4 DNS USER INTERFACE

6.1.4.1 DNS Administration

This document is concerned with design and implementation issues in host software, not with administrative or operational issues. However, administrative issues are of particular importance in the DNS, since errors in particular segments of this large distributed database can cause poor or erroneous performance for many sites. These issues are discussed in [DNS:6] and [DNS:7].

6.1.4.2 DNS User Interface

Hosts MUST provide an interface to the DNS for all application programs running on the host. This interface will typically direct requests to a system process to perform the resolver function [DNS:1, 6.1:2].

At a minimum, the basic interface MUST support a request for all information of a specific type and class associated with a specific name, and it MUST return either all of the requested information, a hard error code, or a soft error indication. When there is no error, the basic interface returns the complete response information without modification, deletion, or ordering, so that the basic interface will not need to be changed to accommodate new data types.

DISCUSSION:
The soft error indication is an essential part of the interface, since it may not always be possible to access particular information from the DNS; see Section 6.1.3.3.

A host MAY provide other DNS interfaces tailored to particular functions, transforming the raw domain data into formats more suited to these functions. In particular, a host MUST provide a DNS interface to facilitate translation between host addresses and host names.

6.1.4.3 Interface Abbreviation Facilities

User interfaces MAY provide a method for users to enter abbreviations for commonly-used names. Although the definition of such methods is outside of the scope of the DNS specification, certain rules are necessary to insure that these methods allow access to the entire DNS name space and to prevent excessive use of Internet resources.

If an abbreviation method is provided, then:

(a) There MUST be some convention for denoting that a name is already complete, so that the abbreviation method(s) are suppressed. A trailing dot is the usual method.

(b) Abbreviation expansion MUST be done exactly once, and MUST be done in the context in which the name was entered.

DISCUSSION:

For example, if an abbreviation is used in a mail program for a destination, the abbreviation should be expanded into a full domain name and stored in the queued message with an indication that it is already complete. Otherwise, the abbreviation might be expanded with a mail system search list, not the user's, or a name could grow due to repeated canonicalizations attempts interacting with wildcards.

The two most common abbreviation methods are:

(1) Interface-level aliases

Interface-level aliases are conceptually implemented as a list of alias/domain name pairs. The list can be per-user or per-host, and separate lists can be associated with different functions, e.g. one list for host name-to-address translation, and a different list for mail domains. When the user enters a name, the interface attempts to match the name to the alias component of a list entry, and if a matching entry can

be found, the name is replaced by the domain name found in the pair.

Note that interface-level aliases and CNAMEs are completely separate mechanisms; interface-level aliases are a local matter while CNAMEs are an Internet-wide aliasing mechanism which is a required part of any DNS implementation.

(2) Search Lists

A search list is conceptually implemented as an ordered list of domain names. When the user enters a name, the domain names in the search list are used as suffixes to the user-supplied name, one by one, until a domain name with the desired associated data is found, or the search list is exhausted. Search lists often contain the name of the local host's parent domain or other ancestor domains. Search lists are often per-user or per-process.

It SHOULD be possible for an administrator to disable a DNS search-list facility. Administrative denial may be warranted in some cases, to prevent abuse of the DNS.

There is danger that a search-list mechanism will generate excessive queries to the root servers while testing whether user input is a complete domain name, lacking a final period to mark it as complete. A search-list mechanism MUST have one of, and SHOULD have both of, the following two provisions to prevent this:

(a) The local resolver/name server can implement caching of negative responses (see Section 6.1.3.3).

(b) The search list expander can require two or more interior dots in a generated domain name before it tries using the name in a query to non-local domain servers, such as the root.

DISCUSSION:
The intent of this requirement is to avoid excessive delay for the user as the search list is tested, and more importantly to prevent excessive traffic to the root and other high-level servers. For example, if the user supplied a name "X" and the search list contained the root as a component,

a query would have to consult a root server before the next search list alternative could be tried. The resulting load seen by the root servers and gateways near the root would be multiplied by the number of hosts in the Internet.

The negative caching alternative limits the effect to the first time a name is used. The interior dot rule is simpler to implement but can prevent easy use of some top-level names.

6.1.5 DOMAIN NAME SYSTEM REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	NOT	FUTURE
GENERAL ISSUES						
Implement DNS name-to-address conversion	6.1.1	x				
Implement DNS address-to-name conversion	6.1.1	x				
Support conversions using host table	6.1.1		x			
Properly handle RR with zero TTL	6.1.2.1	x				
Use QCLASS=* unnecessarily	6.1.2.2		x			
Use QCLASS=IN for Internet class	6.1.2.2	x				
Unused fields zero	6.1.2.3	x				
Use compression in responses	6.1.2.4	x				
Include config info in responses	6.1.2.5				x	
Support all well-known, class-indep. types	6.1.3.5	x				
Easily expand type list	6.1.3.5		x			
Load all RR types (except MD and MF)	6.1.3.6	x				
Load MD or MF type	6.1.3.6				x	
Operate when root servers, etc. unavailable	6.1.3.7	x				
RESOLVER ISSUES:						
Resolver support multiple concurrent requests	6.1.3.1		x			
Full-service resolver:	6.1.3.1			x		
Local caching	6.1.3.1	x				

Information in local cache times out	6.1.3.1	x					
Configurable with starting info	6.1.3.1		x				
Stub resolver:	6.1.3.1			x			
Use redundant recursive name servers	6.1.3.1	x					
Local caching	6.1.3.1			x			
Information in local cache times out	6.1.3.1	x					
Support for remote multi-homed hosts:							
Sort multiple addresses by preference list	6.1.3.4		x				
TRANSPORT PROTOCOLS:							
Support UDP queries	6.1.3.2	x					
Support TCP queries	6.1.3.2		x				
Send query using UDP first	6.1.3.2	x					1
Try TCP if UDP answers are truncated	6.1.3.2		x				
Name server limit TCP query resources	6.1.3.2			x			
Punish unnecessary TCP query	6.1.3.2				x		
Use truncated data as if it were not	6.1.3.2					x	
Private agreement to use only TCP	6.1.3.2			x			
Use TCP for zone transfers	6.1.3.2	x					
TCP usage not block UDP queries	6.1.3.2	x					
Support broadcast or multicast queries	6.1.3.2			x			
RD bit set in query	6.1.3.2					x	
RD bit ignored by server is b'cast/m'cast	6.1.3.2	x					
Send only as occasional probe for addr's	6.1.3.2		x				
RESOURCE USAGE:							
Transmission controls, per [DNS:2]	6.1.3.3	x					
Finite bounds per request	6.1.3.3	x					
Failure after retries => soft error	6.1.3.3	x					
Cache temporary failures	6.1.3.3		x				
Cache negative responses	6.1.3.3		x				
Retries use exponential backoff	6.1.3.3		x				
Upper, lower bounds	6.1.3.3		x				
Client handle Source Quench	6.1.3.3		x				
Server ignore Source Quench	6.1.3.3			x			
USER INTERFACE:							
All programs have access to DNS interface	6.1.4.2	x					
Able to request all info for given name	6.1.4.2	x					
Returns complete info or error	6.1.4.2	x					
Special interfaces	6.1.4.2			x			
Name<->Address translation	6.1.4.2	x					
Abbreviation Facilities:	6.1.4.3			x			

Convention for complete names	6.1.4.3	x					
Conversion exactly once	6.1.4.3	x					
Conversion in proper context	6.1.4.3	x					
Search list:	6.1.4.3			x			
Administrator can disable	6.1.4.3		x				
Prevention of excessive root queries	6.1.4.3	x					
Both methods	6.1.4.3		x				

1. Unless there is private agreement between particular resolver and particular server.

6.2 HOST INITIALIZATION

6.2.1 INTRODUCTION

This section discusses the initialization of host software across a connected network, or more generally across an Internet path. This is necessary for a diskless host, and may optionally be used for a host with disk drives. For a diskless host, the initialization process is called "network booting" and is controlled by a bootstrap program located in a boot ROM.

To initialize a diskless host across the network, there are two distinct phases:

(1) Configure the IP layer.

Diskless machines often have no permanent storage in which to store network configuration information, so that sufficient configuration information must be obtained dynamically to support the loading phase that follows. This information must include at least the IP addresses of the host and of the boot server. To support booting across a gateway, the address mask and a list of default gateways are also required.

(2) Load the host system code.

During the loading phase, an appropriate file transfer protocol is used to copy the system code across the network from the boot server.

A host with a disk may perform the first step, dynamic configuration. This is important for microcomputers, whose floppy disks allow network configuration information to be mistakenly duplicated on more than one host. Also, installation of new hosts is much simpler if they automatically obtain their configuration information from a central server, saving administrator time and decreasing the probability of mistakes.

6.2.2 REQUIREMENTS

6.2.2.1 Dynamic Configuration

A number of protocol provisions have been made for dynamic configuration.

- ICMP Information Request/Reply messages

This obsolete message pair was designed to allow a host to find the number of the network it is on. Unfortunately, it was useful only if the host already knew the host number part of its IP address, information that hosts requiring dynamic configuration seldom had.

- Reverse Address Resolution Protocol (RARP) [BOOT:4]

  RARP is a link-layer protocol for a broadcast medium that allows a host to find its IP address given its link layer address. Unfortunately, RARP does not work across IP gateways and therefore requires a RARP server on every network. In addition, RARP does not provide any other configuration information.

- ICMP Address Mask Request/Reply messages

  These ICMP messages allow a host to learn the address mask for a particular network interface.

- BOOTP Protocol [BOOT:2]

  This protocol allows a host to determine the IP addresses of the local host and the boot server, the name of an appropriate boot file, and optionally the address mask and list of default gateways. To locate a BOOTP server, the host broadcasts a BOOTP request using UDP. Ad hoc gateway extensions have been used to transmit the BOOTP broadcast through gateways, and in the future the IP Multicasting facility will provide a standard mechanism for this purpose.

The suggested approach to dynamic configuration is to use the BOOTP protocol with the extensions defined in "BOOTP Vendor Information Extensions" RFC-1084 [BOOT:3]. RFC-1084 defines some important general (not vendor-specific) extensions. In particular, these extensions allow the address mask to be supplied in BOOTP; we RECOMMEND that the address mask be supplied in this manner.

DISCUSSION:

Historically, subnetting was defined long after IP, and so a separate mechanism (ICMP Address Mask messages) was designed to supply the address mask to a host. However, the IP address mask and the corresponding IP address conceptually form a pair, and for operational

simplicity they ought to be defined at the same time and by the same mechanism, whether a configuration file or a dynamic mechanism like BOOTP.

Note that BOOTP is not sufficiently general to specify the configurations of all interfaces of a multihomed host. A multihomed host must either use BOOTP separately for each interface, or configure one interface using BOOTP to perform the loading, and perform the complete initialization from a file later.

Application layer configuration information is expected to be obtained from files after loading of the system code.

6.2.2.2 Loading Phase

A suggested approach for the loading phase is to use TFTP [BOOT:1] between the IP addresses established by BOOTP.

TFTP to a broadcast address SHOULD NOT be used, for reasons explained in Section 4.2.3.4.

6.3 REMOTE MANAGEMENT

6.3.1 INTRODUCTION

The Internet community has recently put considerable effort into the development of network management protocols. The result has been a two-pronged approach [MGT:1, MGT:6]: the Simple Network Management Protocol (SNMP) [MGT:4] and the Common Management Information Protocol over TCP (CMOT) [MGT:5].

In order to be managed using SNMP or CMOT, a host will need to implement an appropriate management agent. An Internet host SHOULD include an agent for either SNMP or CMOT.

Both SNMP and CMOT operate on a Management Information Base (MIB) that defines a collection of management values. By reading and setting these values, a remote application may query and change the state of the managed system.

A standard MIB [MGT:3] has been defined for use by both management protocols, using data types defined by the Structure of Management Information (SMI) defined in [MGT:2]. Additional MIB variables can be introduced under the "enterprises" and "experimental" subtrees of the MIB naming space [MGT:2].

Every protocol module in the host SHOULD implement the relevant MIB variables. A host SHOULD implement the MIB variables as defined in the most recent standard MIB, and MAY implement other MIB variables when appropriate and useful.

6.3.2 PROTOCOL WALK-THROUGH

The MIB is intended to cover both hosts and gateways, although there may be detailed differences in MIB application to the two cases. This section contains the appropriate interpretation of the MIB for hosts. It is likely that later versions of the MIB will include more entries for host management.

A managed host must implement the following groups of MIB object definitions: System, Interfaces, Address Translation, IP, ICMP, TCP, and UDP.

The following specific interpretations apply to hosts:

- ipInHdrErrors

  Note that the error "time-to-live exceeded" can occur in a host only when it is forwarding a source-routed datagram.

- o ipOutNoRoutes

  This object counts datagrams discarded because no route can be found. This may happen in a host if all the default gateways in the host's configuration are down.

- o ipFragOKs, ipFragFails, ipFragCreates

  A host that does not implement intentional fragmentation (see "Fragmentation" section of [INTRO:1]) MUST return the value zero for these three objects.

- o icmpOutRedirects

  For a host, this object MUST always be zero, since hosts do not send Redirects.

- o icmpOutAddrMaskReps

  For a host, this object MUST always be zero, unless the host is an authoritative source of address mask information.

- o ipAddrTable

  For a host, the "IP Address Table" object is effectively a table of logical interfaces.

- o ipRoutingTable

  For a host, the "IP Routing Table" object is effectively a combination of the host's Routing Cache and the static route table described in "Routing Outbound Datagrams" section of [INTRO:1].

  Within each ipRouteEntry, ipRouteMetric1...4 normally will have no meaning for a host and SHOULD always be -1, while ipRouteType will normally have the value "remote".

  If destinations on the connected network do not appear in the Route Cache (see "Routing Outbound Datagrams section of [INTRO:1]), there will be no entries with ipRouteType of "direct".

DISCUSSION:

The current MIB does not include Type-of-Service in an ipRouteEntry, but a future revision is expected to make

this addition.

We also expect the MIB to be expanded to allow the remote management of applications (e.g., the ability to partially reconfigure mail systems). Network service applications such as mail systems should therefore be written with the "hooks" for remote management.

6.3.3 MANAGEMENT REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Support SNMP or CMOT agent	6.3.1		x				
Implement specified objects in standard MIB	6.3.1		x				

7. REFERENCES

This section lists the primary references with which every implementer must be thoroughly familiar. It also lists some secondary references that are suggested additional reading.

INTRODUCTORY REFERENCES:

[INTRO:1] "Requirements for Internet Hosts -- Communication Layers," IETF Host Requirements Working Group, R. Braden, Ed., RFC-1122, October 1989.

[INTRO:2] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:3] "Official Internet Protocols," J. Reynolds and J. Postel, RFC-1011, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:4] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC-980, March 1986.

[INTRO:5] "Assigned Numbers," J. Reynolds and J. Postel, RFC-1010, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

TELNET REFERENCES:

[TELNET:1] "Telnet Protocol Specification," J. Postel and J. Reynolds, RFC-854, May 1983.

[TELNET:2] "Telnet Option Specification," J. Postel and J. Reynolds, RFC-855, May 1983.

[TELNET:3] "Telnet Binary Transmission," J. Postel and J. Reynolds, RFC-856, May 1983.

[TELNET:4] "Telnet Echo Option," J. Postel and J. Reynolds, RFC-857, May 1983.

[TELNET:5] "Telnet Suppress Go Ahead Option," J. Postel and J.

Reynolds, RFC-858, May 1983.

[TELNET:6] "Telnet Status Option," J. Postel and J. Reynolds, RFC-859, May 1983.

[TELNET:7] "Telnet Timing Mark Option," J. Postel and J. Reynolds, RFC-860, May 1983.

[TELNET:8] "Telnet Extended Options List," J. Postel and J. Reynolds, RFC-861, May 1983.

[TELNET:9] "Telnet End-Of-Record Option," J. Postel, RFC-855, December 1983.

[TELNET:10] "Telnet Terminal-Type Option," J. VanBokkelen, RFC-1091, February 1989.

This document supercedes RFC-930.

[TELNET:11] "Telnet Window Size Option," D. Waitzman, RFC-1073, October 1988.

[TELNET:12] "Telnet Linemode Option," D. Borman, RFC-1116, August 1989.

[TELNET:13] "Telnet Terminal Speed Option," C. Hedrick, RFC-1079, December 1988.

[TELNET:14] "Telnet Remote Flow Control Option," C. Hedrick, RFC-1080, November 1988.

SECONDARY TELNET REFERENCES:

[TELNET:15] "Telnet Protocol," MIL-STD-1782, U.S. Department of Defense, May 1984.

This document is intended to describe the same protocol as RFC-854. In case of conflict, RFC-854 takes precedence, and the present document takes precedence over both.

[TELNET:16] "SUPDUP Protocol," M. Crispin, RFC-734, October 1977.

[TELNET:17] "Telnet SUPDUP Option," M. Crispin, RFC-736, October 1977.

[TELNET:18] "Data Entry Terminal Option," J. Day, RFC-732, June 1977.

[TELNET:19] "TELNET Data Entry Terminal option -- DODIIS Implementation," A. Yasuda and T. Thompson, RFC-1043, February 1988.

FTP REFERENCES:

[FTP:1] "File Transfer Protocol," J. Postel and J. Reynolds, RFC-959, October 1985.

[FTP:2] "Document File Format Standards," J. Postel, RFC-678, December 1974.

[FTP:3] "File Transfer Protocol," MIL-STD-1780, U.S. Department of Defense, May 1984.

This document is based on an earlier version of the FTP specification (RFC-765) and is obsolete.

TFTP REFERENCES:

[TFTP:1] "The TFTP Protocol Revision 2," K. Sollins, RFC-783, June 1981.

MAIL REFERENCES:

[SMTP:1] "Simple Mail Transfer Protocol," J. Postel, RFC-821, August 1982.

[SMTP:2] "Standard For The Format of ARPA Internet Text Messages," D. Crocker, RFC-822, August 1982.

This document obsoleted an earlier specification, RFC-733.

[SMTP:3] "Mail Routing and the Domain System," C. Partridge, RFC-974, January 1986.

This RFC describes the use of MX records, a mandatory extension to the mail delivery process.

[SMTP:4] "Duplicate Messages and SMTP," C. Partridge, RFC-1047, February 1988.

[SMTP:5a] "Mapping between X.400 and RFC 822," S. Kille, RFC-987, June 1986.

[SMTP:5b] "Addendum to RFC-987," S. Kille, RFC-???, September 1987.

The two preceding RFC's define a proposed standard for gatewaying mail between the Internet and the X.400 environments.

[SMTP:6] "Simple Mail Transfer Protocol," MIL-STD-1781, U.S. Department of Defense, May 1984.

This specification is intended to describe the same protocol as does RFC-821. However, MIL-STD-1781 is incomplete; in particular, it does not include MX records [SMTP:3].

[SMTP:7] "A Content-Type Field for Internet Messages," M. Sirbu, RFC-1049, March 1988.

DOMAIN NAME SYSTEM REFERENCES:

[DNS:1] "Domain Names - Concepts and Facilities," P. Mockapetris, RFC-1034, November 1987.

This document and the following one obsolete RFC-882, RFC-883, and RFC-973.

[DNS:2] "Domain Names - Implementation and Specification," RFC-1035, P. Mockapetris, November 1987.

[DNS:3] "Mail Routing and the Domain System," C. Partridge, RFC-974, January 1986.

[DNS:4] "DoD Internet Host Table Specification," K. Harrenstein, RFC-952, M. Stahl, E. Feinler, October 1985.

SECONDARY DNS REFERENCES:

[DNS:5] "Hostname Server," K. Harrenstein, M. Stahl, E. Feinler, RFC-953, October 1985.

[DNS:6] "Domain Administrators Guide," M. Stahl, RFC-1032, November 1987.

[DNS:7] "Domain Administrators Operations Guide," M. Lottor, RFC-1033, November 1987.

[DNS:8] "The Domain Name System Handbook," Vol. 4 of Internet Protocol Handbook, NIC 50007, SRI Network Information Center, August 1989.

SYSTEM INITIALIZATION REFERENCES:

[BOOT:1] "Bootstrap Loading Using TFTP," R. Finlayson, RFC-906, June 1984.

[BOOT:2] "Bootstrap Protocol (BOOTP)," W. Croft and J. Gilmore, RFC-951, September 1985.

[BOOT:3] "BOOTP Vendor Information Extensions," J. Reynolds, RFC-1084, December 1988.

Note: this RFC revised and obsoleted RFC-1048.

[BOOT:4] "A Reverse Address Resolution Protocol," R. Finlayson, T. Mann, J. Mogul, and M. Theimer, RFC-903, June 1984.

MANAGEMENT REFERENCES:

[MGT:1] "IAB Recommendations for the Development of Internet Network Management Standards," V. Cerf, RFC-1052, April 1988.

[MGT:2] "Structure and Identification of Management Information for TCP/IP-based internets," M. Rose and K. McCloghrie, RFC-1065, August 1988.

[MGT:3] "Management Information Base for Network Management of TCP/IP-based internets," M. Rose and K. McCloghrie, RFC-1066, August 1988.

[MGT:4] "A Simple Network Management Protocol," J. Case, M. Fedor, M. Schoffstall, and C. Davin, RFC-1098, April 1989.

[MGT:5] "The Common Management Information Services and Protocol over TCP/IP," U. Warrier and L. Besaw, RFC-1095, April 1989.

[MGT:6] "Report of the Second Ad Hoc Network Management Review Group," V. Cerf, RFC-1109, August 1989.

Security Considerations

There are many security issues in the application and support programs of host software, but a full discussion is beyond the scope of this RFC. Security-related issues are mentioned in sections concerning TFTP (Sections 4.2.1, 4.2.3.4, 4.2.3.5), the SMTP VRFY and EXPN commands (Section 5.2.3), the SMTP HELO command (5.2.5), and the SMTP DATA command (Section 5.2.8).

Author's Address

Robert Braden
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (213) 822 1511

EMail: Braden@ISI.EDU

Appendix D

RFC-1009, Requirements for Internet Gateways

This appendix reproduces, in-full and un-edited, an RFC published by the Internet Engineering Task Force of the Internet Activities Board.

Network Working Group
Request for Comments: 1009
Obsoletes: 985

R. Braden
J. Postel
ISI
June 1987

Requirements for Internet Gateways

Status of this Memo

This document is a formal statement of the requirements to be met by gateways used in the Internet system. As such, it is an official specification for the Internet community. Distribution of this memo is unlimited.

This RFC summarizes the requirements for gateways to be used between networks supporting the Internet protocols. While it was written specifically to support National Science Foundation research programs, the requirements are stated in a general context and are applicable throughout the Internet community.

The purpose of this document is to present guidance for vendors offering gateway products that might be used or adapted for use in an Internet application. It enumerates the protocols required and gives references to RFCs and other documents describing the current specifications. In a number of cases the specifications are evolving and may contain ambiguous or incomplete information. In these cases further discussion giving specific guidance is included in this document. Specific policy issues relevant to the NSF scientific networking community are summarized in an Appendix. As other specifications are updated this document will be revised. Vendors are encouraged to maintain contact with the Internet research community.

1. Introduction

The following material is intended as an introduction and background for those unfamiliar with the Internet architecture and the Internet gateway model. General background and discussion on the Internet architecture and supporting protocol suite can be found in the DDN Protocol Handbook [25] and ARPANET Information Brochure [26], see also [19, 28, 30, 31].

The Internet protocol architecture was originally developed under DARPA sponsorship to meet both military and civilian communication requirements [32]. The Internet system presently supports a variety of government and government-sponsored operational and research activities. In particular, the National Science Foundation (NSF) is building a major extension to the Internet to provide user access to

national supercomputer centers and other national scientific resources, and to provide a computer networking capability to a large number of universities and colleges.

In this document there are many terms that may be obscure to one unfamiliar with the Internet protocols. There is not much to be done about that but to learn, so dive in. There are a few terms that are much abused in general discussion but are carefully and intentionally used in this document. These few terms are defined here.

Packet — A packet is the unit of transmission on a physical network.

Datagram — A datagram is the unit of transmission in the IP protocol. To cross a particular network a datagram is encapsulated inside a packet.

Router — A router is a switch that receives data transmission units from input interfaces and, depending on the addresses in those units, routes them to the appropriate output interfaces. There can be routers at different levels of protocol. For example, Interface Message Processors (IMPs) are packet-level routers.

Gateway — In the Internet documentation generally, and in this document specifically, a gateway is an IP-level router. In the Internet community the term has a long history of this usage [32].

1.1. The DARPA Internet Architecture

1.1.1. Internet Protocols

The Internet system consists of a number of interconnected packet networks supporting communication among host computers using the Internet protocols. These protocols include the Internet Protocol (IP), the Internet Control Message Protocol (ICMP), the Transmission Control Protocol (TCP), and application protocols depending upon them [22].

All Internet protocols use IP as the basic data transport mechanism. IP [1,31] is a datagram, or connectionless, internetwork service and includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security information. ICMP [2] is considered an integral

part of IP, although it is architecturally layered upon IP. ICMP provides error reporting, flow control and first-hop

gateway redirection.

Reliable data delivery is provided in the Internet protocol suite by transport-level protocols such as the Transmission Control Protocol (TCP), which provides end-end retransmission, resequencing and connection control. Transport-level connectionless service is provided by the User Datagram Protocol (UDP).

1.1.2. Networks and Gateways

The constituent networks of the Internet system are required only to provide packet (connectionless) transport. This requires only delivery of individual packets. According to the IP service specification, datagrams can be delivered out of order, be lost or duplicated and/or contain errors. Reasonable performance of the protocols that use IP (e.g., TCP) requires an IP datagram loss rate of less than 5%. In those networks providing connection-oriented service, the extra reliability provided by virtual circuits enhances the end-end robustness of the system, but is not necessary for Internet operation.

Constituent networks may generally be divided into two classes:

* Local-Area Networks (LANs)

  LANs may have a variety of designs, typically based upon buss, ring, or star topologies. In general, a LAN will cover a small geographical area (e.g., a single building or plant site) and provide high bandwidth with low delays.

* Wide-Area Networks (WANs)

  Geographically-dispersed hosts and LANs are interconnected by wide-area networks, also called long-haul networks. These networks may have a complex internal structure of lines and packet-routers (typified by ARPANET), or they may be as simple as point-to-point lines.

In the Internet model, constituent networks are connected together by IP datagram forwarders which are called "gateways" or "IP routers". In this document, every use of the term "gateway" is equivalent to "IP router". In current practice, gateways are normally realized with packet-switching software

executing on a general-purpose CPU, but special-purpose hardware may also be used (and may be required for future higher-throughput gateways).

A gateway is connected to two or more networks, appearing to each of these networks as a connected host. Thus, it has a physical interface and an IP address on each of the connected networks. Forwarding an IP datagram generally requires the gateway to choose the address of the next-hop gateway or (for the final hop) the destination host. This choice, called "routing", depends upon a routing data-base within the gateway. This routing data-base should be maintained dynamically to reflect the current topology of the Internet system; a gateway normally accomplishes this by participating in distributed routing and reachability algorithms with other gateways. Gateways provide datagram transport only, and they seek to minimize the state information necessary to sustain this service in the interest of routing flexibility and robustness.

Routing devices may also operate at the network level; in this memo we will call such devices MAC routers (informally called "level-2 routers", and also called "bridges"). The name derives from the fact that MAC routers base their routing decision on the addresses in the MAC headers; e.g., in IEEE 802.3 networks, a MAC router bases its decision on the 48-bit addresses in the MAC header. Network segments which are connected by MAC routers share the same IP network number, i.e., they logically form a single IP network.

Another variation on the simple model of networks connected with gateways sometimes occurs: a set of gateways may be interconnected with only serial lines, to effectively form a network in which the routing is performed at the internetwork (IP) level rather than the network level.

1.1.3. Autonomous Systems

For technical, managerial, and sometimes political reasons, the gateways of the Internet system are grouped into collections called "autonomous systems" [35]. The gateways included in a single autonomous system (AS) are expected to:

* Be under the control of a single operations and maintenance (O&M) organization;

* Employ common routing protocols among themselves, to maintain their routing data-bases dynamically.

A number of different dynamic routing protocols have been developed (see Section 4.1); the particular choice of routing protocol within a single AS is generically called an interior gateway protocol or IGP.

An IP datagram may have to traverse the gateways of two or more

ASs to reach its destination, and the ASs must provide each other with topology information to allow such forwarding. The Exterior Gateway Protocol (EGP) is used for this purpose, between gateways of different autonomous systems.

1.1.4. Addresses and Subnets

An IP datagram carries 32-bit source and destination addresses, each of which is partitioned into two parts -- a constituent network number and a host number on that network. Symbolically:

```
IP-address ::=  { <Network-number>,  <Host-number> }
```

To finally deliver the datagram, the last gateway in its path must map the host-number (or "rest") part of an IP address into the physical address of a host connection to the constituent network.

This simple notion has been extended by the concept of "subnets", which were introduced in order to allow arbitrary complexity of interconnected LAN structures within an organization, while insulating the Internet system against explosive growth in network numbers and routing complexity. Subnets essentially provide a two-level hierarchical routing structure for the Internet system. The subnet extension, described in RFC-950 [21], is now a required part of the Internet architecture. The basic idea is to partition the <host number> field into two parts: a subnet number, and a true host number on that subnet.

```
IP-address ::=
        { <Network-number>, <Subnet-number>, <Host-number> }
```

The interconnected LANs of an organization will be given the same network number but different subnet numbers. The distinction between the subnets of such a subnetted network must not be visible outside that network. Thus, wide-area routing in the rest of the Internet will be based only upon the <Network-number> part of the IP destination address; gateways outside the network will lump <Subnet-number> and <Host-number>

together to form an uninterpreted "rest" part of the 32-bit IP address. Within the subnetted network, the local gateways must route on the basis of an extended network number:

```
{ <Network-number>, <Subnet-number> }.
```

The bit positions containing this extended network number are indicated by a 32-bit mask called the "subnet mask" [21]; it is

recommended but not required that the <Subnet-number> bits be contiguous and fall between the <Network-number> and the <Host-number> fields. No subnet should be assigned the value zero or -1 (all one bits).

Flexible use of the available address space will be increasingly important in coping with the anticipated growth of the Internet. Thus, we allow a particular subnetted network to use more than one subnet mask. Several campuses with very large LAN configurations are also creating nested hierarchies of subnets, sub-subnets, etc.

There are special considerations for the gateway when a connected network provides a broadcast or multicast capability; these will be discussed later.

1.2. The Internet Gateway Model

There are two basic models for interconnecting local-area networks and wide-area (or long-haul) networks in the Internet. In the first, the local-area network is assigned a network number and all gateways in the Internet must know how to route to that network. In the second, the local-area network shares (a small part of) the address space of the wide-area network. Gateways that support this second model are called "address sharing gateways" or "transparent gateways". The focus of this memo is on gateways that support the first model, but this is not intended to exclude the use of transparent gateways.

1.2.1. Internet Gateways

An Internet gateway is an IP-level router that performs the following functions:

1. Conforms to specific Internet protocols specified in this document, including the Internet Protocol (IP), Internet Control Message Protocol (ICMP), and others as necessary. See Section 2 (Protocols Required).

2. Interfaces to two or more packet networks. For each

connected network the gateway must implement the functions required by that network. These functions typically include:

a. encapsulating and decapsulating the IP datagrams with the connected network framing (e.g., an Ethernet header and checksum);

b. sending and receiving IP datagrams up to the maximum size supported by that network, this size is the

network's "Maximum Transmission Unit" or "MTU";

c. translating the IP destination address into an appropriate network-level address for the connected network (e.g., an Ethernet hardware address);

d. responding to the network flow control and error indication, if any.

See Section 3 (Constituent Network Interface), for details on particular constituent network interfaces.

3. Receives and forwards Internet datagrams. Important issues are buffer management, congestion control, and fairness. See Section 4 (Gateway Algorithms).

a. Recognizes various error conditions and generates ICMP error and information messages as required.

b. Drops datagrams whose time-to-live fields have reached zero.

c. Fragments datagrams when necessary to fit into the MTU of the next network.

4. Chooses a next-hop destination for each IP datagram, based on the information in its routing data-base. See Section 4 (Gateway Algorithms).

5. Supports an interior gateway protocol (IGP) to carry out distributed routing and reachability algorithms with the other gateways in the same autonomous system. In addition, some gateways will need to support the Exterior Gateway Protocol (EGP) to exchange topological information with other autonomous systems. See Section 4 (Gateway Algorithms).

6. Provides system support facilities, including loading, debugging, status reporting, exception reporting and control. See Section 5 (Operation and Maintenance).

1.2.2. Embedded Gateways

A gateway may be a stand-alone computer system, dedicated to its IP router functions. Alternatively, it is possible to embed gateway functionality within a host operating system which supports connections to two or more networks. The best-known example of an operating system with embedded gateway code is the Berkeley BSD system. The embedded gateway feature

seems to make internetting easy, but it has a number of hidden pitfalls:

1. If a host has only a single constituent-network interface, it should not act as a gateway.

   For example, hosts with embedded gateway code that gratuitously forward broadcast packets or datagrams on the same net often cause packet avalanches.

2. If a (multihomed) host acts as a gateway, it must implement ALL the relevant gateway requirements contained in this document.

   For example, the routing protocol issues (see Sections 2.6 and 4.1) and the control and monitoring problems are as hard and important for embedded gateways as for stand-alone gateways.

   Since Internet gateway requirements and specifications may change independently of operating system changes, an administration that operates an embedded gateway in the Internet is strongly advised to have an ability to maintain and update the gateway code (e.g., this might require gateway code source).

3. Once a host runs embedded gateway code, it becomes part of the Internet system. Thus, errors in software or configuration of such a host can hinder communication between other hosts. As a consequence, the host administrator must lose some autonomy.

   In many circumstances, a host administrator will need to disable gateway coded embedded in the operating system, and any embedded gateway code must be organized so it can be easily disabled.

4. If a host running embedded gateway code is concurrently used for other services, the O&M (operation and maintenance) requirements for the two modes of use may be in serious conflict.

   For example, gateway O&M will in many cases be performed remotely by an operations center; this may require privileged system access which the host administrator would not normally want to distribute.

1.2.3. Transparent Gateways

The basic idea of a transparent gateway is that the hosts on the local-area network behind such a gateway share the address

space of the wide-area network in front of the gateway. In certain situations this is a very useful approach and the limitations do not present significant drawbacks.

The words "in front" and "behind" indicate one of the limitations of this approach: this model of interconnection is suitable only for a geographically (and topologically) limited stub environment. It requires that there be some form of logical addressing in the network level addressing of the wide-area network (that is, all the IP addresses in the local environment map to a few (usually one) physical address in the wide-area network, in a way consistent with the { IP address <-> network address } mapping used throughout the wide-area network).

Multihoming is possible on one wide-area network, but may present routing problems if the interfaces are geographically or topologically separated. Multihoming on two (or more) wide-area networks is a problem due to the confusion of addresses.

The behavior that hosts see from other hosts in what is apparently the same network may differ if the transparent gateway cannot fully emulate the normal wide-area network service. For example, if there were a transparent gateway between the ARPANET and an Ethernet, a remote host would not receive a Destination Dead message [3] if it sent a datagram to an Ethernet host that was powered off.

1.3. Gateway Characteristics

Every Internet gateway must perform the functions listed above. However, a vendor will have many choices on power, complexity, and features for a particular gateway product. It may be helpful to observe that the Internet system is neither homogeneous nor fully-connected. For reasons of technology and geography, it is growing into a global-interconnect system plus a "fringe" of LANs around the "edge".

* The global-interconnect system is comprised of a number of wide-area networks to which are attached gateways of several ASs; there are relatively few hosts connected directly to it. The global-interconnect system includes the ARPANET, the NSFNET "backbone", the various NSF regional and consortium networks, other ARPA sponsored networks such as

the SATNET and the WBNET, and the DCA sponsored MILNET. It is anticipated that additional networks sponsored by these and other agencies (such as NASA and DOE) will join the global-interconnect system.

* Most hosts are connected to LANs, and many organizations have clusters of LANs interconnected by local gateways. Each such cluster is connected by gateways at one or more points into the global-interconnect system. If it is connected at only one point, a LAN is known as a "stub" network.

Gateways in the global-interconnect system generally require:

* Advanced routing and forwarding algorithms

  These gateways need routing algorithms which are highly dynamic and also offer type-of-service routing. Congestion is still not a completely resolved issue [24]. Improvements to the current situation will be implemented soon, as the research community is actively working on these issues.

* High availability

  These gateways need to be highly reliable, providing 24 hour a day, 7 days a week service. In case of failure, they must recover quickly.

* Advanced O&M features

  These gateways will typically be operated remotely from a regional or national monitoring center. In their

interconnect role, they will need to provide sophisticated means for monitoring and measuring traffic and other events and for diagnosing faults.

* High performance

  Although long-haul lines in the Internet today are most frequently 56 Kbps, DS1 lines (1.5 Mbps) are of increasing importance, and even higher speeds are likely in the future. Full-duplex operation is provided at any of these speeds.

  The average size of Internet datagrams is rather small, of the order of 100 bytes. At DS1 line speeds, the per-datagram processing capability of the gateways, rather than the line speed, is likely to be the bottleneck. To fill a DS1 line with average-sized Internet datagrams, a gateway would need to pass -- receive, route, and send -- 2,000 datagrams per second per interface. That is, a

gateway which supported 3 DS1 lines and and Ethernet interface would need to be able to pass a dazzling 2,000 datagrams per second in each direction on each of the interfaces, or a aggregate throughput of 8,000 datagrams per second, in order to fully utilize DS1 lines. This is beyond the capability of current gateways.

Note: some vendors count input and output operations separately in datagrams per second figures; for these vendors, the above example would imply 16,000 datagrams per second !

Gateways used in the "LAN fringe" (e.g., campus networks) will generally have to meet less stringent requirements for performance, availability, and maintenance. These may be high or medium-performance devices, probably competitively procured from several different vendors and operated by an internal organization (e.g., a campus computing center). The design of these gateways should emphasize low average delay and good burst performance, together with delay and type-of-service sensitive resource management. In this environment, there will be less formal O&M, more hand-crafted static configurations for special cases, and more need for inter-operation with gateways of other vendors. The routing mechanism will need to be very flexible, but need not be so highly dynamic as in the global-interconnect system.

It is important to realize that Internet gateways normally operate in an unattended mode, but that equipment and software faults can have a wide-spread (sometimes global) effect. In any environment,

a gateway must be highly robust and able to operate, possibly in a degraded state, under conditions of extreme congestion or failure of network resources.

Even though the Internet system is not fully-interconnected, many parts of the system do need to have redundant connectivity. A rich connectivity allows reliable service despite failures of communication lines and gateways, and it can also improve service by shortening Internet paths and by providing additional capacity. The engineering tradeoff between cost and reliability must be made for each component of the Internet system.

2. Protocols Required in Gateways

The Internet architecture uses datagram gateways to interconnect constituent networks. This section describes the various protocols which a gateway needs to implement.

2.1. Internet Protocol (IP)

IP is the basic datagram protocol used in the Internet system [19, 31]. It is described in RFC-791 [1] and also in MIL-STD-1777 [5] as clarified by RFC-963 [36] ([1] and [5] are intended to describe the same standard, but in quite different words). The subnet extension is described in RFC-950 [21].

With respect to current gateway requirements the following IP features can be ignored, although they may be required in the future: Type of Service field, Security option, and Stream ID option. However, if recognized, the interpretation of these quantities must conform to the standard specification.

It is important for gateways to implement both the Loose and Strict Source Route options. The Record Route and Timestamp options are useful diagnostic tools and must be supported in all gateways.

The Internet model requires that a gateway be able to fragment datagrams as necessary to match the MTU of the network to which they are being forwarded, but reassembly of fragmented datagrams is generally left to the destination hosts. Therefore, a gateway will not perform reassembly on datagrams it forwards.

However, a gateway will generally receive some IP datagrams addressed to itself; for example, these may be ICMP Request/Reply messages, routing update messages (see Sections 2.3 and 2.6), or for monitoring and control (see Section 5). For these datagrams, the gateway will be functioning as a destination host, so it must implement IP reassembly in case the datagrams have been fragmented by some transit gateway. The destination gateway must have a reassembly buffer which is at least as large as the maximum of the MTU values for its network interfaces and 576. Note also that it is possible for a particular protocol implemented by a host or gateway to require a lower bound on reassembly buffer size which is larger than 576. Finally, a datagram which is addressed to a gateway may use any of that gateway’s IP addresses as destination address, regardless of which interface the datagram enters.

There are five classes of IP addresses: Class A through Class E [23]. Of these, Class D and Class E addresses are

reserved for experimental use. A gateway which is not participating in these experiments must ignore all datagrams with a Class D or Class E destination IP address. ICMP Destination Unreachable or ICMP Redirect messages must not result from receiving such datagrams.

There are certain special cases for IP addresses, defined in the latest Assigned Numbers document [23]. These special cases can be concisely summarized using the earlier notation for an IP address:

IP-address ::= { <Network-number>, <Host-number> }

or

IP-address ::= { <Network-number>, <Subnet-number>, <Host-number> }

if we also use the notation "-1" to mean the field contains all 1 bits. Some common special cases are as follows:

(a) {0, 0}

This host on this network. Can only be used as a source address (see note later).

(b) {0, <Host-number>}

Specified host on this network. Can only be used as a source address.

(c) { -1, -1}

Limited broadcast. Can only be used as a destination address, and a datagram with this address must never be forwarded outside the (sub-)net of the source.

(d) {<Network-number>, -1}

Directed broadcast to specified network. Can only be used as a destination address.

(e) {<Network-number>, <Subnet-number>, -1}

Directed broadcast to specified subnet. Can only be used as a destination address.

(f) {<Network-number>, -1, -1}

RFC 1009 - Requirements for Internet Gateways June 1987

Directed broadcast to all subnets of specified subnetted network. Can only be used as a destination address.

(g) {127, <any>}

Internal host loopback address. Should never appear outside a host.

The following two are conventional notation for network numbers, and do not really represent IP addresses. They can never be used in an IP datagram header as an IP source or destination address.

(h) {<Network-number>, 0}

Specified network (no host).

(i) {<Network-number>, <Subnet-number>, 0}

Specified subnet (no host).

Note also that the IP broadcast address, which has primary application to Ethernets and similar technologies that support an inherent broadcast function, has an all-ones value in the host field of the IP address. Some early implementations chose the all-zeros value for this purpose, which is not in conformance with the specification [23, 49, 50].

2.2. Internet Control Message Protocol (ICMP)

ICMP is an auxiliary protocol used to convey advice and error messages and is described in RFC-792 [2].

We will discuss issues arising from gateway handling of particular ICMP messages. The ICMP messages are grouped into two classes: error messages and information messages. ICMP error messages are never sent about ICMP error messages, nor about broadcast or multicast datagrams.

The ICMP error messages are: Destination Unreachable, Redirect, Source Quench, Time Exceeded, and Parameter Problem.

The ICMP information messages are: Echo, Information, Timestamp, and Address Mask.

2.2.1. Destination Unreachable

The distinction between subnets of a subnetted network, which depends on the address mask described in RFC-950 [21], must not be visible outside that network. This distinction is important in the case of the ICMP Destination Unreachable message.

The ICMP Destination Unreachable message is sent by a gateway in response to a datagram which it cannot forward because the destination is unreachable or down. The gateway chooses one of the following two types of Destination Unreachable messages to send:

* Net Unreachable

* Host Unreachable

Net unreachable implies that an intermediate gateway was unable to forward a datagram, as its routing data-base gave no next hop for the datagram, or all paths were down. Host Unreachable implies that the destination network was reachable, but that a gateway on that network was unable to reach the destination host. This might occur if the particular destination network was able to determine that the desired host was unreachable or down. It might also occur when the destination host was on a subnetted network and no path was available through the subnets of this network to the destination. Gateways should send Host Unreachable messages whenever other hosts on the same destination network might be reachable; otherwise, the source host may erroneously conclude that ALL hosts on the network are unreachable, and that may not be the case.

2.2.2. Redirect

The ICMP Redirect message is sent by a gateway to a host on the same network, in order to change the gateway used by the host for routing certain datagrams. A choice of four types of Redirect messages is available to specify datagrams destined for a particular host or network, and possibly with a particular type-of-service.

If the directly-connected network is not subnetted, a gateway can normally send a network Redirect which applies to all hosts on a specified remote network. Using a network rather than a host Redirect may economize slightly on network traffic and on host routing table storage. However, the saving is not significant, and subnets create an ambiguity about the subnet

mask to be used to interpret a network Redirect. In a general subnet environment, it is difficult to specify precisely the cases in which network Redirects can be used.

Therefore, it is recommended that a gateway send only host (or host and type-of-service) Redirects.

2.2.3. Source Quench

All gateways must contain code for sending ICMP Source Quench messages when they are forced to drop IP datagrams due to congestion. Although the Source Quench mechanism is known to be an imperfect means for Internet congestion control, and research towards more effective means is in progress, Source Quench is considered to be too valuable to omit from production gateways.

There is some argument that the Source Quench should be sent before the gateway is forced to drop datagrams [62]. For example, a parameter X could be established and set to have Source Quench sent when only X buffers remain. Or, a parameter Y could be established and set to have Source Quench sent when only Y per cent of the buffers remain.

Two problems for a gateway sending Source Quench are: (1) the consumption of bandwidth on the reverse path, and (2) the use of gateway CPU time. To ameliorate these problems, a gateway must be prepared to limit the frequency with which it sends Source Quench messages. This may be on the basis of a count (e.g., only send a Source Quench for every N dropped datagrams overall or per given source host), or on the basis of a time (e.g., send a Source Quench to a given source host or overall at most once per T millseconds). The parameters (e.g., N or T) must be settable as part of the configuration of the gateway; furthermore, there should be some configuration setting which disables sending Source Quenches. These configuration parameters, including disabling, should ideally be specifiable separately for each network interface.

Note that a gateway itself may receive a Source Quench as the result of sending a datagram targeted to another gateway. Such datagrams might be an EGP update, for example.

2.2.4. Time Exceeded

The ICMP Time Exceeded message may be sent when a gateway discards a datagram due to the TTL being reduced to zero. It

may also be sent by a gateway if the fragments of a datagram addressed to the gateway itself cannot be reassembled before the time limit.

2.2.5. Parameter Problem

The ICMP Parameter Problem message may be sent to the source host for any problem not specifically covered by another ICMP message.

2.2.6. Address Mask

Host and gateway implementations are expected to support the ICMP Address Mask messages described in RFC-950 [21].

2.2.7. Timestamp

The ICMP Timestamp message has proven to be useful for diagnosing Internet problems. The preferred form for a timestamp value, the "standard value", is in milliseconds since midnight GMT. However, it may be difficult to provide this value with millisecond resolution. For example, many systems use clocks which update only at line frequency, 50 or 60 times per second. Therefore, some latitude is allowed in a "standard" value:

* The value must be updated at a frequency of at least 30 times per second (i.e., at most five low-order bits of the value may be undefined).

* The origin of the value must be within a few minutes of midnight, i.e., the accuracy with which operators customarily set CPU clocks.

To meet the second condition for a stand-alone gateway, it will be necessary to query some time server host when the gateway is booted or restarted. It is recommended that the UDP Time Server Protocol [44] be used for this purpose. A more advanced implementation would use NTP (Network Time Protocol) [45] to achieve nearly millisecond clock synchronization; however, this is not required.

Even if a gateway is unable to establish its time origin, it ought to provide a "non-standard" timestamp value (i.e., with the non-standard bit set), as a time in milliseconds from system startup.

New gateways, especially those expecting to operate at T1 or higher speeds, are expected to have at least millisecond clocks.

2.2.8. Information Request/Reply

The Information Request/Reply pair was intended to support self-configuring systems such as diskless workstations, to

allow them to discover their IP network numbers at boot time. However, the Reverse ARP (RARP) protocol [15] provides a better mechanism for a host to use to discover its own IP address, and RARP is recommended for this purpose. Information Request/Reply need not be implemented in a gateway.

2.2.9. Echo Request/Reply

A gateway must implement ICMP Echo, since it has proven to be an extremely useful diagnostic tool. A gateway must be prepared to receive, reassemble, and echo an ICMP Echo Request datagram at least as large as the maximum of 576 and the MTU's of all of the connected networks. See the discussion of IP reassembly in gateways, Section 2.1.

The following rules resolve the question of the use of IP source routes in Echo Request and Reply datagrams. Suppose a gateway D receives an ICMP Echo Request addressed to itself from host S.

1. If the Echo Request contained no source route, D should send an Echo Reply back to S using its normal routing rules. As a result, the Echo Reply may take a different path than the Request; however, in any case, the pair will sample the complete round-trip path which any other higher-level protocol (e.g., TCP) would use for its data and ACK segments between S and D.

2. If the Echo Request did contain a source route, D should send an Echo Reply back to S using as a source route the return route built up in the source-routing option of the Echo Request.

2.3. Exterior Gateway Protocol (EGP)

EGP is the protocol used to exchange reachability information between Autonomous Systems of gateways, and is defined in RFC-904 [11]. See also RFC-827 [51], RFC-888 [46], and RFC-975 [27] for background information. The most widely used EGP implementation is described in RFC-911 [13].

When a dynamic routing algorithm is operated in the gateways of an Autonomous System (AS), the routing data-base must be coupled to

the EGP implementation. This coupling should ensure that, when a net is determined to be unreachable by the routing algorithm, the net will not be declared reachable to other ASs via EGP. This requirement is designed to minimize spurious traffic to "black holes" and to ensure fair utilization of the resources on other systems.

The present EGP specification defines a model with serious limitations, most importantly a restriction against propagating "third party" EGP information in order to prevent long-lived routing loops [27]. This effectively limits EGP to a two-level hierarchy; the top level is formed by the "core" AS, while the lower level is composed of those ASs which are direct neighbor gateways to the core AS. In practice, in the current Internet, nearly all of the "core gateways" are connected to the ARPANET, while the lower level is composed of those ASs which are directly gatewayed to the ARPANET or MILNET.

RFC-975 [27] suggested one way to generalize EGP to lessen these topology restrictions; it has not been adopted as an official specification, although its ideas are finding their way into the new EGP developments. There are efforts underway in the research community to develop an EGP generalization which will remove these restrictions.

In EGP, there is no standard interpretation (i.e., metric) for the distance fields in the update messages, so distances are comparable only among gateways of the same AS. In using EGP data, a gateway should compare the distances among gateways of the same AS and prefer a route to that gateway which has the smallest distance value.

The values to be announced in the distance fields for particular networks within the local AS should be a gateway configuration parameter; by suitable choice of these values, it will be possible to arrange primary and backup paths from other AS's. There are other EGP parameters, such as polling intervals, which also need to be set in the gateway configuration.

When routing updates become large they must be transmitted in parts. One strategy is to use IP fragmentation, another is to explicitly send the routing information in sections. The Internet Engineering Task Force is currently preparing a recommendation on this and other EGP engineering issues.

2.4. Address Resolution Protocol (ARP)

ARP is an auxiliary protocol used to perform dynamic address translation between LAN hardware addresses and Internet addresses, and is described in RFC-826 [4].

ARP depends upon local network broadcast. In normal ARP usage, the initiating host broadcasts an ARP Request carrying a target IP address; the corresponding target host, recognizing its own IP address, sends back an ARP Reply containing its own hardware interface address.

A variation on this procedure, called "proxy ARP", has been used by gateways attached to broadcast LANs [14]. The gateway sends an ARP Reply specifying its interface address in response to an ARP Request for a target IP address which is not on the directly-connected network but for which the gateway offers an appropriate route. By observing ARP and proxy ARP traffic, a gateway may accumulate a routing data-base [14].

Proxy ARP (also known in some quarters as "promiscuous ARP" or "the ARP hack") is useful for routing datagrams from hosts which do not implement the standard Internet routing rules fully -- for example, host implementations which predate the introduction of subnetting. Proxy ARP for subnetting is discussed in detail in RFC-925 [14].

Reverse ARP (RARP) allows a host to map an Ethernet interface address into an IP address [15]. RARP is intended to allow a self-configuring host to learn its own IP address from a server at boot time.

2.5. Constituent Network Access Protocols

See Section 3.

2.6. Interior Gateway Protocols

Distributed routing algorithms continue to be the subject of research and engineering, and it is likely that advances will be made over the next several years. A good algorithm needs to respond rapidly to real changes in Internet connectivity, yet be stable and insensitive to transients. It needs to synchronize the distributed data-base across gateways of its Autonomous System rapidly (to avoid routing loops), while consuming only a small fraction of the available bandwidth.

Distributed routing algorithms are commonly broken down into the following three components:

A. An algorithm to assign a "length" to each Internet path.

The "length" may be a simple count of hops (1, or infinity if the path is broken), or an administratively-assigned cost, or some dynamically-measured cost (usually an average delay).

In order to determine a path length, each gateway must at least test whether each of its neighbors is reachable; for this purpose, there must be a "reachability" or "neighbor up/down" protocol.

B. An algorithm to compute the shortest path(s) to a given destination.

C. A gateway-gateway protocol used to exchange path length and routing information among gateways.

The most commonly-used IGPs in Internet gateways are as follows.

2.6.1. Gateway-to-Gateway Protocol (GGP)

GGP was designed and implemented by BBN for the first experimental Internet gateways [41]. It is still in use in the BBN LSI/11 gateways, but is regarded as having serious drawbacks [58]. GGP is based upon an algorithm used in the early ARPANET IMPs and later replaced by SPF (see below).

GGP is a "min-hop" algorithm, i.e., its length measure is simply the number of network hops between gateway pairs. It implements a distributed shortest-path algorithm, which requires global convergence of the routing tables after a change in topology or connectivity. Each gateway sends a GGP

routing update only to its neighbors, but each update includes an entry for every known network, where each entry contains the hop count from the gateway sending the update.

2.6.2. Shortest-Path-First (SPF) Protocols

SPF [40] is the name for a class of routing algorithms based on a shortest-path algorithm of Dijkstra. The current ARPANET routing algorithm is SPF, and the BBN Butterfly gateways also use SPF. Its characteristics are considered superior to GGP [58].

Under SPF, the routing data-base is replicated rather than distributed. Each gateway will have its own copy of the same data-base, containing the entire Internet topology and the lengths of every path. Since each gateway has all the routing

data and runs a shortest-path algorithm locally, there is no problem of global convergence of a distributed algorithm, as in GGP. To build this replicated data-base, a gateway sends SPF routing updates to ALL other gateways; these updates only list the distances to each of the gateway's neighbors, making them much smaller than GGP updates. The algorithm used to distribute SPF routing updates involves reliable flooding.

2.6.3. Routing Information (RIP)

RIP is the name often used for a class of routing protocols based upon the Xerox PUP and XNS routing protocols. These are relatively simple, and are widely available because they are incorporated in the embedded gateway code of Berkeley BSD systems. Because of this simplicity, RIP protocols have come the closest of any to being an "Open IGP", i.e., a protocol which can be used between different vendors' gateways. Unfortunately, there is no standard, and in fact not even a good document, for RIP.

As in GGP, gateways using RIP periodically broadcast their routing data-base to their neighbor gateways, and use a hop-count as the metric.

A fixed value of the hop-count (normally 16) is defined to be "infinity", i.e., network unreachable. A RIP implementation must include measures to avoid both the slow-convergence phenomen called "counting to infinity" and the formation of routing loops. One such measure is a "hold-down" rule. This rule establishes a period of time (typically 60 seconds) during which a gateway will ignore new routing information about a given network, once the gateway has learned that network is

unreachable (has hop-count "infinity"). The hold-down period must be settable in the gateway configuration; if gateways with different hold-down periods are using RIP in the same Autonomous System, routing loops are a distinct possibility. In general, the hold-down period is chosen large enough to allow time for unreachable status to propagate to all gateways in the AS.

2.6.4. Hello

The "Fuzzball" software for an LSI/11 developed by Dave Mills incorporated an IGP called the "Hello" protocol [39]. This IGP is mentioned here because the Fuzzballs have been widely used in Internet experimentation, and because they have served as a testbed for many new routing ideas.

2.7. Monitoring Protocols

See Section 5 of this document.

2.8. Internet Group Management Protocol (IGMP)

An extension to the IP protocol has been defined to provide Internet-wide multicasting, i.e., delivery of copies of the same IP datagram to a set of Internet hosts [47, 48]. This delivery is to be performed by processes known as "multicasting agents", which reside either in a host on each net or (preferably) in the gateways.

The set of hosts to which a datagram is delivered is called a "host group", and there is a host-agent protocol called IGMP, which a host uses to join, leave, or create a group. Each host group is distinguished by a Class D IP address.

This multicasting mechanism and its IGMP protocol are currently experimental; implementation in vendor gateways would be premature at this time. A datagram containing a Class D IP address must be dropped, with no ICMP error message.

3. Constituent Network Interface

This section discusses the rules used for transmission of IP datagrams on the most common types of constituent networks. A gateway must be able to send and receive IP datagrams of any size up to the MTU of any constituent network to which it is connected.

3.1. Public data networks via X.25

The formats specified for public data networks accessed via X.25 are described in RFC-877 [8]. Datagrams are transmitted over standard level-3 virtual circuits as complete packet sequences. Virtual circuits are usually established dynamically as required and time-out after a period of no traffic. Link-level retransmission, resequencing and flow control are performed by the network for each virtual circuit and by the LAPB link-level protocol. Note that a single X.25 virtual circuit may be used to multiplex all IP traffic between a pair of hosts. However, multiple parallel virtual circuits may be used in order to improve the utilization of the subscriber access line, in spite of small

X.25 window sizes; this can result in random resequencing.

The correspondence between Internet and X.121 addresses is usually established by table-lookup. It is expected that this will be replaced by some sort of directory procedure in the future. The table of the hosts on the Public Data Network is in the Assigned Numbers [23].

The normal MTU is 576; however, the two DTE's (hosts or gateways) can use X.25 packet size negotiation to increase this value [8].

3.2. ARPANET via 1822 LH, DH, or HDH

The formats specified for ARPANET networks using 1822 access are described in BBN Report 1822 [3], which includes the procedures for several subscriber access methods. The Distant Host (DH) method is used when the host and IMP (the Defense Communication Agency calls it a Packet Switch Node or PSN) are separated by not more than about 2000 feet of cable, while the HDLC Distant Host (HDH) is used for greater distances where a modem is required. Under HDH, retransmission, resequencing and flow control are performed by the network and by the HDLC link-level protocol.

The IP encapsulation format is simply to include the IP datagram as the data portion of an 1822 message. In addition, the high-order 8 bits of the Message Id field (also known as the "link" field") should be set to 155 [23]. The MTU is 1007 octets.

While the ARPANET 1822 protocols are widely used at present, they are expected to be eventually overtaken by the DDN Standard X.25 protocol (see Section 3.3). The original IP address mapping (RFC-796 [38]) is in the process of being replaced by a new interface specification called AHIP-E; see RFC-1005 [61] for the proposal.

Gateways connected to ARPANET or MILNET IMPs using 1822 access must incorporate features to avoid host-port blocking (i.e., RFNM counting) and to detect and report as ICMP Unreachable messages the failure of destination hosts or gateways (i.e., convert the 1822 error messages to the appropriate ICMP messages).

In the development of a network interface it will be useful to review the IMP end-to-end protocol described in RFC-979 [29].

3.3. ARPANET via DDN Standard X.25

The formats specified for ARPANET networks via X.25 are described in the Defense Data Network X.25 Host Interface Specification [6], which describes two sets of procedures: the DDN Basic X.25, and the DDN Standard X.25. Only DDN Standard X.25 provides the

functionality required for interoperability assumptions of the Internet protocol.

The DDN Standard X.25 procedures are similar to the public data network X.25 procedures, except in the address mappings. Retransmission, resequencing and flow control are performed by the network and by the LAPB link-level protocol. Multiple parallel virtual circuits may be used in order to improve the utilization of the subscriber access line; this can result in random resequencing.

Gateways connected to ARPANET or MILNET using Standard X.25 access must detect and report as ICMP Unreachable messages the failure of destination hosts or gateways (i.e., convert the X.25 diagnostic codes to the appropriate ICMP messages).

To achieve compatibility with 1822 interfaces, the effective MTU for a Standard X.25 interface is 1007 octets.

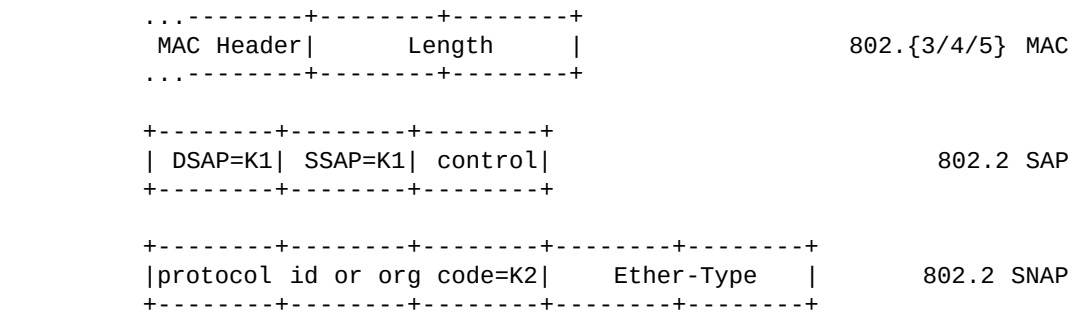
3.4. Ethernet and IEEE 802

The formats specified for Ethernet networks are described in RFC-894 [10]. Datagrams are encapsulated as Ethernet packets with 48-bit source and destination address fields and a 16-bit type field (the type field values are listed in the Assigned Numbers [23]). Address translation between Ethernet addresses and Internet addresses is managed by the Address Resolution Protocol, which is required in all Ethernet implementations. There is no explicit link-level retransmission, resequencing or flow control, although most hardware interfaces will retransmit automatically in case of collisions on the cable.

The IEEE 802 networks use a Link Service Access Point (LSAP) field in much the same way the ARPANET uses the "link" field. Further, there is an extension of the LSAP header called the Sub-Network Access Protocol (SNAP).

The 802.2 encapsulation is used on 802.3, 802.4, and 802.5 network by using the SNAP with an organization code indicating that the following 16 bits specify the Ether-Type code [23].

Headers:



The total length of the SAP Header and the SNAP header is 8-octets, making the 802.2 protocol overhead come out on a 64-bit boundary.

K1 is 170. The IEEE likes to talk about things in bit transmission order and specifies this value as 01010101. In big-endian order, as used in the Internet specifications, this becomes 10101010 binary, or AA hex, or 170 decimal. K2 is 0 (zero).

The use of the IP LSAP (K1 = 6) is reserved for future development.

The assigned values for the Ether-Type field are the same for either this IEEE 802 encapsulation or the basic Ethernet encapsulation [10].

In either Ethernets or IEEE 802 nets, the IP datagram is the data portion of the packet immediately following the Ether-Type.

The MTU for an Ethernet or its IEEE-standard equivalent (802.3) is 1500 octets.

3.5. Serial-Line Protocols

In some configurations, gateways may be interconnected with each other by means of serial asynchronous or synchronous lines, with or without modems. When justified by the expected error rate and other factors, a link-level protocol may be required on the serial line. While there is no single Internet standard for this protocol, it is suggested that one of the following protocols be used.

* X.25 LAPB (Synchronous Lines)

  This is the link-level protocol used for X.25 network access. It includes HDLC "bit-stuffing" as well as rotating-window flow control and reliable delivery.

A gateway must be configurable to play the role of either the DCE or the DTE.

* HDLC Framing (Synchronous Lines)

  This is just the bit-stuffing and framing rules of LAPB. It is the simplest choice, although it provides no flow control or reliable delivery; however, it does provide error detection.

* Xerox Synchronous Point-to-Point (Synchronous Lines)

  This Xerox protocol is an elaboration upon HDLC framing that includes negotiation of maximum packet sizes, dial-up or dedicated circuits, and half- or full-duplex operation [12].

* Serial Line Framing Protocol (Asynchronous Lines)

  This protocol is included in the MIT PC/IP package for an IBM PC and is defined in Appendix I to the manual for that system [20].

It will be important to make efficient use of the bandwidth available on a serial line between gateways. For example, it is desirable to provide some form of data compression. One possible standard compression algorithm, "Thinwire II", is described in RFC-914 [42]. This and similar algorithms are tuned to the particular types of redundancy which occur in IP and TCP headers; however, more work is necessary to define a standard serial-line compression protocol for Internet gateways. Until a standard has been adopted, each vendor is free to choose a compression algorithm; of course, the result will only be useful on a serial line between two gateways using the same compression algorithm.

Another way to ensure maximum use of the bandwidth is to avoid unnecessary retransmissions at the link level. For some kinds of IP traffic, low delay is more important than reliable delivery. The serial line driver could distinguish such datagrams by their IP TOS field, and place them on a special high-priority, no-retransmission queue.

A serial point-to-point line between two gateways may be considered to be a (particularly simple) network, a "null net". Considered in this way, a serial line requires no special considerations in the routing algorithms of the connected gateways, but does need an IP network number. To avoid the wholesale consumption of Internet routing data-base space by null nets, we strongly recommend that subnetting be used for null net numbering, whenever possible.

For example, assume that network 128.203 is to be constructed of gateways joined by null nets; these nets are given (sub-)net numbers 128.203.1, 128.203.2, etc., and the two interfaces on each end of null net 128.203.s might have IP addresses 128.203.s.1 and 128.203.s.2.

An alternative model of a serial line is that it is not a network, but rather an internal communication path joining two "half gateways". It is possible to design an IGP and routing algorithm that treats a serial line in this manner [39, 52].

4. Gateway Algorithms

Gateways are general packet-switches that forward packets according to the IP address, i.e., they are IP routers. While it is beyond the scope of this document to specify the details of the mechanisms used in any particular, perhaps proprietary, gateway architecture, there are a number of basic algorithms which must be provided by any acceptable design.

4.1. Routing Algorithm

The routing mechanism is fundamental to Internet operation. In all but trivial network topologies, robust Internet service requires some degree of routing dynamics, whether it be effected by manual or automatic means or by some combination of both. In particular, if routing changes are made manually, it must be possible to make these routing changes from a remote Network Operation Center (NOC) without taking down the gateway for reconfiguration. If static routes are used, there must be automatic fallback or rerouting features.

Handling unpredictable changes in Internet connectivity must be considered the normal case, so that systems of gateways will normally be expected to have a routing algorithm with the capability of reacting to link and other gateway failures and changing the routing automatically.

This document places no restriction on the type of routing algorithm, e.g., node-based, link-based or any other algorithm, or on the routing distance metric, e.g., delay or hop-count.

However, the following features are considered necessary for a successful gateway routing algorithm:

1. The algorithm must sense the failure or restoration of a link or other gateway and switch to appropriate paths. A design objective is to switch paths within an interval less than the typical TCP user time-out (one minute is a safe assumption).

2. The algorithm must suppress routing loops between neighbor gateways and must contain provisions to avoid or suppress routing loops that may form between non-neighbor gateways. A design objective is for no loop to persist for longer than an interval greater than the typical TCP user time-out.

3. The control traffic necessary to operate the routing algorithm must not significantly degrade or disrupt normal

network operation. Changes in state which might momentarily disrupt normal operation in a local-area must not cause disruption in remote areas of the network.

4. As the size of the network increases, the demand on resources must be controlled in an efficient way. Table lookups should be hashed, for example, and data-base updates handled piecemeal, with only incremental changes broadcast over a wide-area.

5. The size of the routing data-base must not be allowed to exceed a constant, independent of network topology, times the number of nodes times the mean connectivity (average number of incident links). An advanced design might not require that the entire routing data-base be kept in any particular gateway, so that discovery and caching techniques would be necessary.

6. Reachability and delay metrics, if used, must not depend on direct connectivity to all other gateways or on the use of network-specific broadcast mechanisms. Polling procedures (e.g., for consistency checking) must be used only sparingly and in no case introduce an overhead exceeding a constant, independent of network topology, times the longest non-looping path.

7. Default routes (generally intended as a means to reduce the size of the routing data-base) must be used with care, because of the many problems with multiple paths, loops, and mis-configurations which routing defaults have caused.

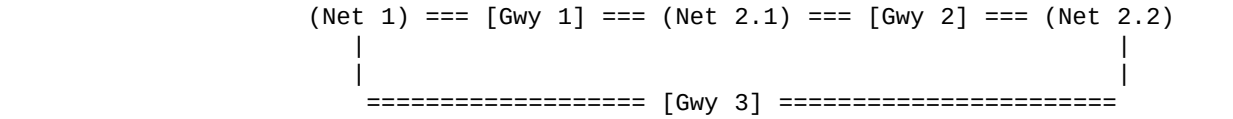
   The most common application of defaults is for routing

within an Internet region which is connected in a strictly hierarchical fashion and is a stub from the rest of the Internet system. In this case, the default is used for routing "up" the tree. Unfortunately, such restricted topology seldom lasts very long, and defaults cease to work.

More generally, defaults could be used for initial routing guesses, with final routes to be discovered and cached from external or internal data-bases via the routing algorithm or EGP.

4.2. Subnets and Routing

We will call a gateway "subnetted" if at least one of its interfaces is connected to a subnet; the set of gateways directly connected to subnets of the same network will be referred to as a "subnet cluster". For example, in the following diagram, network 2 is subnetted, with subnets 2.1 and 2.2, but network 1 is not; gateways 1, 2, and 3 are subnetted and are members of the same subnet cluster.



Subnets have the following effects on gateway routing:

A. Non-subnetted gateways are not affected at all.

B. The routing data-base in a subnetted gateway must consider the address mask for subnet entries.

C. Routing updates among the gateways in the same subnet cluster must include entries for the various subnets. The corresponding address mask(s) may be implicit, but for full generality the mask needs to be given explicitly for each entry. Note that if the routing data-base included a full 32-bit mask for every IP network, the gateway could deal with networks and subnets in a natural way. This would also handle the case of multiple subnet masks for the same subnetted network.

D. Routing updates from a subnetted gateway to a gateway outside the cluster can contain nets, never subnets.

E. If a subnetted gateway (e.g., gateway 2 above) is unable to forward a datagram from one subnet to another subnet of the same network, then it must return a Host Unreachable, not a Net Unreachable, as discussed in Section 2.2.1.

When considering the choice of routing protocol, a gateway builder must consider how that protocol generalizes for subnets. For some routing protocols it will be possible to use the same procedures in a regular gateway and a subnetted gateway, with only a change of parameters (e.g., address masks).

A different subnet address mask must be configurable for each

interface of a given gateway. This will allow a subnetted gateway to connect to two different subnetted networks, or to connect two subnets of the same network with different masks.

4.3 Resource Allocation

In order to perform its basic datagram-forwarding functions, a gateway must allocate resources; its packet buffers and CPU time must be allocated to packets it receives from connected networks, while the bandwidth to each of the networks must also be allocated for sending packets. The choice of allocation strategies will be critical when a particular resource is scarce. The most obvious allocation strategy, first-come-first-served (FCFS), may not be appropriate under overload conditions, for reasons which we will now explore.

A first example is buffer allocation. It is important for a gateway to allocate buffers fairly among all of its connected networks, even if these networks have widely varying bandwidths. A high-speed interface must not be allowed to starve slower interfaces of buffers. For example, consider a gateway with a 10 Mbps Ethernet connection and two 56 Kbps serial lines. A buggy host on the Ethernet may spray that gateway interface with packets at high speed. Without careful algorithm design in the gateway, this could tie up all the gateway buffers in such a way that transit traffic between the serial lines would be completely stopped.

Allocation of output bandwidth may also require non-FCFS strategies. In an advanced gateway design, allocation of output bandwidth may depend upon Type-of-Service bits in the IP headers. A gateway may also want to give priority to datagrams for its own up/down and routing protocols.

Finally, Nagle [24] has suggested that gateways implement "fair queueing", i.e., sharing output bandwidth equitably among the current traffic sources. In his scheme, for each network

interface there would be a dynamically-built set of output queues, one per IP source address; these queues would be serviced in a round-robin fashion to share the bandwidth. If subsequent research shows fair queueing to be desirable, it will be added to a future version of this document as a universal requirement.

4.4. Special Addresses and Filters

Section 2.1 contained a list of the 32-bit IP addresses which have special meanings. They do not in general represent unique IP addresses of Internet hosts, and there are restrictions on their use in IP headers.

We can distinguish two classes of these special cases. The first class (specifically, cases (a), (b), (c), (g), (h), and (i) in section 2.1) contains addresses which should never appear in the destination address field of any IP datagram, so a gateway should never be asked to route to one of these addresses. However, in the real world of imperfect implementations and configuration errors, such bad destination addresses do occur. It is the responsibility of a gateway to avoid propagating such erroneous addresses; this is especially important for gateways included in the global interconnect system. In particular, a gateway which receives a datagram with one of these forbidden addresses should:

1. Avoid inserting that address into its routing database, and avoid including it in routing updates to any other gateway.

2. Avoid forwarding a datagram containing that address as a destination.

To enforce these restrictions, it is suggested that a gateway include a configurable filter for datagrams and routing updates. A typical filter entry might consist of a 32-bit mask and value pair. If the logical AND of the given address with the mask equals the value, a match has been found. Since filtering will consume gateway resources, it is vital that the gateway configuration be able to control the degree of filtering in use.

There is a second class of special case addresses (cases (d), (e), and (f) in section 2.1), the so-called "directed broadcasts". A directed broadcast is a datagram to be forwarded normally to the specified destination (sub-)net and then broadcast on the final hop. An Internet gateway is permitted, but not required, to filter out directed broadcasts destined for any of its

locally-connected networks. Hence, it should be possible to configure the filter to block the delivery of directed broadcasts.

Finally, it will also be useful for Internet O&M to have a configurable filter on the IP source address. This will allow a network manager to temporarily block traffic from a particular misbehaving host, for example.

4.5. Redirects

The ICMP Redirect message is specified only for use by a gateway to update the routing table of a host on the same connected net. However, the Redirect message is sometimes used between gateways, due to the following considerations:

> The routing function in a host is very much like that in a "dumb gateway" (i.e., a gateway having only static routes). It is desirable to allow the routing tables of a dumb gateway to be changed under the control of a dynamic gateway (i.e., a gateway with full dynamic routing) on the same network. By analogy, it is natural to let the dynamic gateway send ICMP Redirect messages to dumb gateway.

The use of ICMP Redirect between gateways in this fashion may be considered to be part of the IGP (in fact, the totality of the IGP, as far as the dumb gateway is concerned!) in the particular Autonomous System. Specification of an IGP is outside the scope of this document, so we only note the possibility of using Redirect in this fashion. Gateways are not required to receive and act upon redirects, and in fact dynamic gateways must ignore them. We also note that considerable experience shows that dumb gateways often create problems resulting in "black holes"; a full routing gateway is always preferable.

Routing table entries established by redirect messages must be removed automatically, either by a time-out or when a use count goes to zero.

4.6. Broadcast and Multicast

A host which is connected to a network (generally a LAN) with an intrinsic broadcast capability may want to use this capability to effect multidestination delivery of IP datagrams. The basic Internet model assumes point-to-point messages, and we must take some care when we incorporate broadcasting. It is important to note that broadcast addresses may occur at two protocol levels: the local network header and the IP header.

Incorrect handling of broadcasting has often been the cause of

packet avalanches (sometimes dubbed "meltdown") in LANs. These avalanches are generally caused by gratuitous datagram-forwarding by hosts, or by hosts sending ICMP error messages when they discard broadcast datagrams.

Gateways have a responsibility to prevent avalanches, or datagrams which can trigger avalanches, from escaping into another network.

In general, a gateway must not forward a datagram which arrives via local network broadcast, and must not send an ICMP error message when dropping the datagram. A discussion of the rules will be found in Appendix A; see also [50].

As noted in Section 4.4, a gateway is permitted to filter out directed broadcasts. Hence, directed broadcasts will only be useful in limited Internet regions (e.g., the within the subnets of a particular campus) in which delivery is supported by the gateway administrators. Host group multicasting (see Sections 2.8 and 4.6) will soon provide a much more efficient mechanism than directed broadcasting. Gateway algorithms for host group multicasting will be specified in future RFC's.

4.7. Reachability Procedures

The architecture must provide a robust mechanism to establish the operational status of each link and node in the network, including the gateways, the links connecting them and, where appropriate, the hosts as well. Ordinarily, this requires at least a link-level reachability protocol involving a periodic exchange of messages across each link. This function might be intrinsic to the link-level protocols used (e.g., LAPB). However, it is in general ill-advised to assume a host or gateway is operating correctly even if its link-level reachability protocol is operating correctly. Additional confirmation is required in the form of an operating routing algorithm or peer-level reachability protocol (such as used in EGP).

Failure and restoration of a link and/or gateway are considered network events and must be reported to the control center. It is desirable, although not required, that reporting paths not require correct functioning of the routing algorithm itself.

4.8. Time-To-Live

The Time-to-Live (TTL) field of the IP header is defined to be a timer limiting the lifetime of a datagram in the Internet. It is an 8-bit field and the units are seconds. This would imply that for a maximum TTL of 255 a datagram would time-out after about 4 and a quarter minutes. Another aspect of the definition requires each gateway (or other module) that handles a datagram to decrement the TTL by at least one, even if the elapsed time was

much less than a second. Since this is very often the case, the TTL effectively becomes a hop count limit on how far a datagram can propagate through the Internet.

As the Internet grows, the number of hops needed to get from one edge to the opposite edge increases, i.e., the Internet diameter grows.

If a gateway holds a datagram for more than one second, it must decrement the TTL by one for each second.

If the TTL is reduced to zero, the datagram must be discarded, and the gateway may send an ICMP Time Exceeded message to the source. A datagram should never be received with a TTL of zero.

When it originates a datagram, a gateway is acting in the role of a host and must supply a realistic initial value for the TTL.

5. Operation and Maintenance

5.1. Introduction

Facilities to support operation and maintenance (O&M) activities form an essential part of any gateway implementation. The following kinds of activity are included under gateway O&M:

* Diagnosing hardware problems in the gateway processor, in its network interfaces, or in the connected networks, modems, or communication lines.

* Installing a new version of the gateway software.

* Restarting or rebooting a gateway after a crash.

* Configuring (or reconfiguring) the gateway.

* Detecting and diagnosing Internet problems such as congestion, routing loops, bad IP addresses, black holes, packet avalanches, and misbehaved hosts.

* Changing network topology, either temporarily (e.g., to diagnose a communication line problem) or permanently.

* Monitoring the status and performance of the gateways and the connected networks.

* Collecting traffic statistics for use in (Inter-)network planning.

Gateways, packet-switches, and their connected communication lines are often operated as a system by a centralized O&M organization. This organization will maintain a (Inter-)network operation center, or NOC, to carry out its O&M functions. It is essential that gateways support remote control and monitoring from such a NOC, through an Internet path (since gateways might not be connected to the same network as their NOC). Furthermore, an IP datagram traversing the Internet will often use gateways under the control of more than one NOC; therefore, Internet problem diagnosis will often involve cooperation of personnel of more than one NOC. In some cases, the same gateway may need to be monitored by more than one NOC.

The tools available for monitoring at a NOC may cover a wide range of sophistication. Proposals have included multi-window, dynamic displays of the entire gateway system, and the use of AI techniques for automatic problem diagnosis.

Gateway O&M facilities discussed here are only a part of the large and difficult problem of Internet management. These problems encompass not only multiple management organizations, but also multiple protocol layers. For example, at the current stage of evolution of the Internet architecture, there is a strong coupling between host TCP implementations and eventual IP-level congestion in the gateway system [9]. Therefore, diagnosis of congestion problems will sometimes require the monitoring of TCP statistics in hosts. Gateway algorithms also interact with local network performance, especially through handling of broadcast packets and ARP, and again diagnosis will require access to hosts (e.g., examining ARP caches). However, consideration of host monitoring is beyond the scope of this RFC.

There are currently a number of R&D efforts in progress in the area of Internet management and more specifically gateway O&M. It is hoped that these will lead quickly to Internet standards for the gateway protocols and facilities required in this area. This is also an area in which vendor creativity can make a significant contribution.

5.2. Gateway O&M Models

There is a range of possible models for performing O&M functions on a gateway. At one extreme is the local-only model, under which the O&M functions can only be executed locally, e.g., from a terminal plugged into the gateway machine. At the other extreme, the fully-remote model allows only an absolute minimum of functions to be performed locally (e.g., forcing a boot), with most O&M being done remotely from the NOC. There intermediate models, e.g., one in which NOC personnel can log into the gateway as a host, using the Telnet protocol, to perform functions which can also be invoked locally. The local-only model may be adequate in a few gateway installations, but in general remote operation from a NOC will be required, and therefore remote O&M provisions are required for most gateways.

Remote O&M functions may be exercised through a control agent (program). In the direct approach, the gateway would support remote O&M functions directly from the NOC using standard Internet protocols (e.g., UDP or TCP); in the indirect approach, the control agent would support these protocols and control the gateway itself using proprietary protocols. The direct approach is preferred, although either approach is acceptable. The use of specialized host hardware and/or software requiring significant additional investment is discouraged; nevertheless, some vendors may elect to provide the control agent as an integrated part of the network in which the gateways are a part. If this is the

RFC 1009 - Requirements for Internet Gateways June 1987

case, it is required that a means be available to operate the control agent from a remote site using Internet protocols and paths and with equivalent functionality with respect to a local agent terminal.

It is desirable that a control agent and any other NOC software tools which a vendor provides operate as user programs in a standard operating system. The use of the standard Internet protocols UDP and TCP for communicating with the gateways should facilitate this.

Remote gateway monitoring and (especially) remote gateway control present important access control problems which must be addressed. Care must also be taken to ensure control of the use of gateway resources for these functions. It is not desirable to let gateway monitoring take more than some limited fraction of the gateway CPU time, for example. On the other hand, O&M functions must receive priority so they can be exercised when the gateway is congested, i.e., when O&M is most needed.

There are no current Internet standards for the control and monitoring protocols, although work is in progress in this area. The Host Monitoring Protocol (HMP) [7] could be used as a model until a standard is developed; however, it is strongly recommended that gateway O&M protocol be built on top of one of the standard Internet end-to-end protocols UDP or TCP. An example of a very simple but effective approach to gateway monitoring is contained in RFC-996 [43].

5.3. Gateway O&M Functions

The following O&M functions need to be performed in a gateway:

A. Maintenance -- Hardware Diagnosis

Each gateway must operate as a stand-alone device for the purposes of local hardware maintenance. Means must be available to run diagnostic programs at the gateway site using only on-site tools, which might be only a diskette or tape and local terminal. It is desirable, although not required, to be able to run diagnostics or dump the gateway via the network in case of fault. Means should be provided to allow remote control from the NOC of of modems attached to the gateway. The most important modem control capability is entering and leaving loopback mode, to diagnose line problems.

B. Control -- Dumping and Rebooting

It must be possible to dump and reboot a stand-alone gateway upon command from the NOC. In addition, a stand-alone gateway must include a watchdog timer that either initiates a reboot automatically or signals a remote control site if not reset periodically by the software. It is desirable that the boot data involved reside at an Internet host (e.g., the NOC host) and be transmitted via the net; however, the use of local devices at the gateway site is acceptable.

C. Control -- Configuring the Gateway

Every gateway will have a number of configuration parameters which must be set (see the next section for examples). It must be possible to update the parameters without rebooting the gateway; at worst, a restart may be required.

D. Monitoring -- Status and Performance

A mechanism must be provided for retrieving status and statistical information from a gateway. A gateway must supply such information in response to a polling message from the NOC. In addition, it may be desirable to configure a gateway to transmit status spontaneously and periodically to a NOC (or set of NOCs), for recording and display.

Examples of interesting status information include: link status, queue lengths, buffer availability, CPU and memory utilization, the routing data-base, error counts, and packet counts. Counts should be kept for dropped datagrams, separated by reason. Counts of ICMP datagrams should be kept by type and categorized into those originating at the gateway, and those destined for the gateway. It would be useful to maintain many of these statistics by network interface, by source/destination network pair, and/or by source/destination host pair.

Note that a great deal of useful monitoring data is often to be found in the routing data-base. It is therefore useful to be able to tap into this data-base from the NOC.

E. Monitoring -- Error Logging

A gateway should be capable of asynchronously sending exception ("trap") reports to one or more specified Internet addresses, one of which will presumably be the NOC host.

There must also be a mechanism to limit the frequency of such trap reports, and the parameters controlling this frequency must be settable in the gateway configuration.

Examples of conditions which should result in traps include: datagrams discarded because of TTL expiration (an indicator of possible routing loops); resource shortages; or an interface changing its up/down status.

5.4. Gateway Configuration Parameters

Every gateway will have a set of configuration parameters controlling its operation. It must be possible to set these parameters remotely from the NOC or locally at any time, without taking the gateway down.

The following is a partial but representative list of possible configuration parameters for a full-function gateway. The items marked with "(i)" should be settable independently for each network interface.

* (i) IP (sub-) network address
* (i) Subnet address mask
* (i) MTU of local network
* (i) Hardware interface address
* (i) Broadcast compatibility option (0s or 1s)
* EGP parameters -- neighbors, Autonomous System number, and polling parameters
* Static and/or default routes, if any
* Enable/Disable Proxy ARP
* Source Quench parameters
* Address filter configuration
* Boot-host address
* IP address of time server host
* IP address(es) of logging host(s)

* IP address(es) of hosts to receive traps
* IP address(es) of hosts authorized to issue control commands
* Error level for logging

* Maximum trap frequency

* Hold-down period (if any)

Appendix A. Technical Details

This Appendix collects a number of technical details and rules concerning datagram forwarding by gateways and datagram handling by hosts, especially in the presence of broadcasting and subnets.

A.1. Rules for Broadcasting

The following rules define how to handle broadcasts of packets and

datagrams [50]:

a. Hosts (which do not contain embedded gateways) must NEVER forward any datagrams received from a connected network, broadcast or not.

   When a host receives an IP datagram, if the destination address identifies the host or is an IP broadcast address, the host passes the datagram to its appropriate higher-level protocol module (possibly sending ICMP protocol unreachable, but not if the IP address was a broadcast address). Any other IP datagram must simply be discarded, without an ICMP error message. Hosts never send redirects.

b. All packets containing IP datagrams which are sent to the local-network packet broadcast address must contain an IP broadcast address as the destination address in their IP header. Expressed in another way, a gateway (or host) must not send in a local-network broadcast packet an IP datagram that has a specific IP host address as its destination field.

c. A gateway must never forward an IP datagram that arrives addressed to the IP limited broadcast address {-1,-1}. Furthermore, it must must not send an ICMP error message about discarding such a datagram.

d. A gateway must not forward an IP datagram addressed to network zero, i.e., {0, *}.

e. A gateway may forward a directed broadcast datagram, i.e., a datagram with the IP destination address:

   { <Network-number>, -1}.

   However, it must not send such a directed broadcast out the same interface it came in, if this interface has <Network-number> as its network number. If the code in the

   gateway making this decision does not know what interface the directed-broadcast datagram arrived on, the gateway cannot support directed broadcast to this connected network at all.

f. A gateway is permitted to protect its connected networks by discarding directed broadcast datagrams.

A gateway will broadcast an IP datagram on a connected network if it is a directed broadcast destined for that network. Some gateway-gateway routing protocols (e.g., RIP) also require

broadcasting routing updates on the connected networks. In either case, the datagram must have an IP broadcast address as its destination.

Note: as observed earlier, some host implementations (those based on Berkeley 4.2BSD) use zero rather than -1 in the host field. To provide compatibility during the period until these systems are fixed or retired, it may be useful for a gateway to be configurable to send either choice of IP broadcast address and accept both if received.

A.2. ICMP Redirects

A gateway will generate an ICMP Redirect if and only if the destination IP address is reachable from the gateway (as determined by the routing algorithm) and the next-hop gateway is on the same (sub-)network as the source host. Redirects must not be sent in response to an IP network or subnet broadcast address or in response to a Class D or Class E IP address.

A host must discard an ICMP Redirect if the destination IP address is not its own IP address, or the new target address is not on the same (sub-)network. An accepted Redirect updates the routing data-base for the old target address. If there is no route associated with the old target address, the Redirect is ignored. If the old route is associated with a default gateway, a new route associated with the new target address is inserted in the data-base.

Appendix B. NSFNET Specific Requirements

The following sections discuss certain issues of special concern to the NSF scientific networking community. These issues have primary relevance in the policy area, but also have ramifications in the technical area.

B.1. Proprietary and Extensibility Issues

Although hosts, gateways and networks supporting Internet technology have been in continuous operation for several years, vendors users and operators must understand that not all networking issues are fully resolved. As a result, when new needs

or better solutions are developed for use in the NSF networking community, it may be necessary to field new protocols or augment existing ones. Normally, these new protocols will be designed to interoperate in all practical respects with existing protocols; however, occasionally it may happen that existing systems must be upgraded to support these new or augmented protocols.

NSF systems procurements may favor those vendors who undertake a commitment to remain aware of current Internet technology and be prepared to upgrade their products from time to time as appropriate. As a result, vendors are strongly urged to consider extensibility and periodic upgrades as fundamental characteristics of their products. One of the most productive and rewarding ways to do this on a long-term basis is to participate in ongoing Internet research and development programs in partnership with the academic community.

B.2. Interconnection Technology

In order to ensure network-level interoperability of different vendor's gateways within the NSFNET context, we specify that a gateway must at a minimum support Ethernet connections and serial line protocol connections.

Currently the most important common interconnection technology between Internet systems of different vendors is Ethernet. Among the reasons for this are the following:

1. Ethernet specifications are well-understood and mature.

2. Ethernet technology is in almost all aspects vendor independent.

3. Ethernet-compatible systems are common and becoming more so.

These advantages combined favor the use of Ethernet technology as the common point of demarcation between NSF network systems supplied by different vendors, regardless of technology. It is a requirement of NSF gateways that, regardless of the possibly proprietary switching technology used to implement a given vendor-supplied network, its gateways must support an Ethernet attachment to gateways of other vendors.

It is expected that future NSF gateway requirements will specify other interconnection technologies. The most likely candidates are those based on X.25 or IEEE 802, but other technologies including broadband cable, optical fiber, or other media may also be considered.

B.3. Routing Interoperability

The Internet does not currently have an "open IGP" standard, i.e., a common IGP which would allow gateways from different vendors to form a single Autonomous System. Several approaches to routing interoperability are currently in use among vendors and the NSF networking community.

* Proprietary IGP

  At least one gateway vendor has implemented a proprietary IGP and uses EGP to interface to the rest of the Internet.

* RIP

  Although RIP is undocumented and various implementations of it differ in subtle ways, it has been used successfully for interoperation among multiple vendors as an IGP.

* Gateway Daemon

  The NSF networking community has built a "gateway daemon" program which can mediate among multiple routing protocols to create a mixed-IGP Autonomous System. In particular, the prototype gateway daemon executes on a 4.3BSD machine acting as a gateway and exchanges routing information with other gateways, speaking both RIP and Hello protocols; in addition, it supports EGP to other Autonomous Systems.

B.4. Multi-Protocol Gateways

The present NSF gateway requirements specify only the Internet protocol IP. However, in a few years the Internet will begin a gradual transition to the functionally-equivalent subset of the ISO protocols [17]. In particular, an increasing percentage of the traffic will use the ISO Connectionless Mode Network Service (CLNS, but commonly called "ISO IP") [33] in place of IP. It is expected that the ISO suite will eventually become the dominant one; however, it is also expected that requirements to support Internet IP will continue, perhaps indefinitely.

To support the transition to ISO protocols and the coexistence stage, it is highly desirable that a gateway design provide for future extensions to support more than one protocol simultaneous, and in particular both IP and CLNS [18].

Present NSF gateway requirements do not include protocols above

the network layer, such as TCP, unless necessary for network monitoring or control. Vendors should recognize that future requirements to interwork between Internet and ISO applications, for example, may result in an opportunity to market gateways supporting multiple protocols at all levels up through the application level [16]. It is expected that the network-level NSF gateway requirements summarized in this document will be incorporated in the requirements document for these application-level gateways.

Internet gateways function as intermediate systems (IS) with respect to the ISO connectionless network model and incorporate defined packet formats, routing algorithms and related procedures [33, 34]. The ISO ES-IS [37] provides the functions of ARP and ICMP Redirect.

B.5. Access Control and Accounting

There are no requirements for NSF gateways at this time to incorporate specific access-control and accounting mechanisms in the design; however, these important issues are currently under study and will be incorporated into a subsequent edition of this document. Vendors are encouraged to plan for the introduction of these mechanisms into their products. While at this time no definitive common model for access control and accounting has emerged, it is possible to outline some general features such a model is likely to have, among them the following:

1. The primary access control and accounting mechanisms will be in the service hosts themselves, not the gateways, packet-switches or workstations.

2. Agents acting on behalf of access control and accounting mechanisms may be necessary in the gateways, to collect data, enforce password protection, or mitigate resource priority and fairness. However, the architecture and protocols used by these agents may be a local matter and cannot be specified in advance.

3. NSF gateways may be required to incorporate access control and accounting mechanisms based on datagram source/destination address, as well as other fields in the IP header.

4. NSF gateways may be required to enforce policies on access to gateway and communication resources. These policies may be based upon equity ("fairness") or upon inequity ("priority").

Acknowledgments

An earlier version of this document (RFC-985) [60] was prepared by Dave Mills in behalf of the Gateway Requirements Subcommittee of the NSF Network Technical Advisory Group, in cooperation with the Internet Activities Board, Internet Architecture Task Force, and Internet Engineering Task Force. This effort was chaired by Dave Mills, and contributed to by many people.

The authors of current document have also received assistance from many people in the NSF and ARPA networking community. We thank you, one and all.

References and Bibliography

Many of these references are available from the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025 (telephone: 800-235-3155).

[1] Postel, J., "Internet Protocol", RFC-791, USC Information Sciences Institute, September 1981.

[2] Postel, J., "Internet Control Message Protocol", RFC-792, USC Information Sciences Institute, September 1981.

[3] BBN, "Interface Message Processor - Specifications for the Interconnection of a Host and an IMP", Report 1822, Bolt Beranek and Newman, December 1981.

[4] Plummer, D., "An Ethernet Address Resolution Protocol", RFC-826, Symbolics, September 1982.

[5] DOD, "Military Standard Internet Protocol", Military Standard MIL-STD-1777, United States Department of Defense, August 1983.

[6] BBN, "Defense Data Network X.25 Host Interface Specification", Report 5476, Bolt Beranek and Newman, December 1983.

[7] Hinden, R., "A Host Monitoring Protocol", RFC-869, BBN Communications, December 1983.

[8] Korb, J.T., "A Standard for the Transmission of IP Datagrams over Public Data Networks", RFC-877, Purdue University, September 1983.

[9] Nagle, J., "Congestion Control in IP/TCP Internetworks", RFC-896, Ford Aerospace, January 1984.

[10] Hornig, C., "A Standard for the Transmission of IP Datagrams over Ethernet Networks", RFC-894, Symbolics, April 1984.

[11] Mills, D.L., "Exterior Gateway Formal Specification", RFC-904, M/A-COM Linkabit, April 1984.

[12] Xerox, "Xerox Synchronous Point-to-Point Protocol", Xerox System Integration Standard 158412, December 1984.

[13] Kirton, P., "EGP Gateway under Berkeley UNIX 4.2", RFC-911, USC Information Sciences Institute, August 1984.

[14] Postel, J., "Multi-LAN Address Resolution", RFC-925, USC Information Sciences Institute, October 1984.

[15] Finlayson, R., T. Mann, J. Mogul, and M. Theimer, "A Reverse Address Resolution Protocol", RFC-904, Stanford University, June 1984.

[16] NRC, "Transport Protocols for Department of Defense Data Networks", RFC-942, National Research Council, March 1985.

[17] Postel, J., "DOD Statement on NRC Report", RFC-945, USC Information Sciences Institute, April 1985.

[18] ISO, "Addendum to the Network Service Definition Covering Network Layer Addressing", RFC-941, International Standards Organization, April 1985.

[19] Leiner, B., J. Postel, R. Cole and D. Mills, "The DARPA Internet Protocol Suite", Proceedings INFOCOM 85, IEEE, Washington DC, March 1985. Also in: IEEE Communications Magazine, March 1985. Also available as ISI-RS-85-153.

[20] Romkey, J., "PC/IP Programmer's Manual", MIT Laboratory for Computer Science, pp. 57-59, April 1986.

[21] Mogul, J., and J. Postel, "Internet Standard Subnetting Procedure", RFC-950, Stanford University, August 1985.

[22] Reynolds, J., and J. Postel, "Official Internet Protocols", RFC-1011, USC Information Sciences Institute, May 1987.

[23] Reynolds, J., and J. Postel, "Assigned Numbers", RFC-1010, USC Information Sciences Institute, May 1987.

[24] Nagle, J., "On Packet Switches with Infinite Storage", RFC-970, Ford Aerospace, December 1985.

[25] SRI, "DDN Protocol Handbook", NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[26] SRI, "ARPANET Information Brochure", NIC-50003, SRI International, December 1985.

[27] Mills, D.L., "Autonomous Confederations", RFC-975, M/A-COM Linkabit, February 1986.

[28] Jacobsen, O., and J. Postel, "Protocol Document Order Information", RFC-980, SRI International, March 1986.

[29] Malis, A.G., "PSN End-to-End Functional Specification", RFC-979, BBN Communications, March 1986.

[30] Postel, J, "Internetwork Applications using the DARPA Protocol Suite", Proceedings INFOCOM 85, IEEE, Washington DC, March 1985. Also available as ISI-RS-85-151.

[31] Postel, J, C. Sunshine, and D. Cohen, "The ARPA Internet Protocol", Computer Networks, Vol. 5, No. 4, July 1981.

[32] Cerf, V., and R. Kahn, "A Protocol for Packet Network Intercommunication", IEEE Transactions on Communication, May 1974.

[33] ISO, "Protocol for Providing the Connectionless-mode Network Service", RFC-994, DIS-8473, International Standards Organization, March 1986.

[34] ANSI, "Draft Network Layer Routing Architecture", ANSI X3S3.3, 86-215R, April 1987.

[35] Rosen, E., "Exterior Gateway Protocol (EGP)", RFC-827, Bolt Beranek and Newman, October 1982.

[36] Sidhu, D., "Some Problems with the Specification of the Military Standard Internet Protocol", RFC-963, Iowa State University, November 1985.

[37] ISO, "End System to Intermediate System Routing Exchange Protocol for use in conjunction with ISO 8473", RFC-995, April 1986.

[38] Postel, J., "Address Mappings", RFC-796, USC/Information Sciences Institute, September 1981.

[39] Mills, D., "DCN Local Network Protocols", RFC-891, M/A-COM Linkabit, December 1983.

[40] McQuillan, J. M., I. Richer, and E. C. Rosen, "The New Routing Algorithm for the ARPANET", IEEE Transactions on Communications, May 1980.

[41] Hinden, R., and A. Sheltzer, "The DARPA Internet Gateway", RFC-823, Bolt Beranek and Newman, September 1982.

[42] Farber, D., G. Delp, and T. Conte, "A Thinwire Protocol for Connecting Personal Computers to the Internet", RFC-914, University of Delaware, September 1984.

[43] Mills, D., "Statistics Server", RFC-996, University Of Delaware, February 1987.

[44] Postel, J. and K. Harrenstien, "Time Protocol", RFC-868, May 1983.

[45] Mills, D., "Network Time Protocol (NTP)", RFC-958, M/A-Com Linkabit, September 1985.

[46] Seamonson, L., and E. Rosen, "Stub Exterior Gateway Protocol", RFC-888, Bolt Beranek And Newman, January 1984.

[47] Deering, S., and D. Cheriton, "Host Groups: A Multicast Extension to the Internet Protocol", RFC-966, Stanford University, December 1985.

[48] Deering, S., "Host Extensions for IP Multicasting", RFC-988, Stanford University, July 1986.

[49] Mogul, J., "Broadcasting Internet Datagrams", RFC-919, Stanford University, October 1984.

[50] Mogul, J., "Broadcasting Internet Datagrams in the Presence of Subnets", RFC-922, Stanford University, October 1984.

[51] Rosen, E., "Exterior Gateway Protocol", RFC-827, Bolt Beranek and Newman, October 1982.

[52] Rose, M., "Low Tech Connection into the ARPA Internet: The Raw Packet Split Gateway", Technical Report 216, Department of Information and Computer Science, University of California, Irvine, February 1984.

[53] Rosen, E., "Issues in Buffer Management", IEN-182, Bolt Beranek and Newman, May 1981.

[54] Rosen, E., "Logical Addressing", IEN-183, Bolt Beranek and Newman, May 1981.

[55] Rosen, E., "Issues in Internetting - Part 1: Modelling the Internet", IEN-184, Bolt Beranek and Newman, May 1981.

[56] Rosen, E., "Issues in Internetting - Part 2: Accessing the Internet", IEN-187, Bolt Beranek and Newman, June 1981.

[57] Rosen, E., "Issues in Internetting - Part 3: Addressing", IEN-188, Bolt Beranek and Newman, June 1981.

[58] Rosen, E., "Issues in Internetting - Part 4: Routing", IEN-189, Bolt Beranek and Newman, June 1981.

[59] Sunshine, C., "Comments on Rosen's Memos", IEN-191, USC Information Sciences Institute, July 1981.

[60] NTAG, "Requirements for Internet Gateways -- Draft", RFC-985, Network Technical Advisory Group, National Science Foundation,

May 1986.

[61] Khanna, A., and Malis, A., "The ARPANET AHIP-E Host Access Protocol (Enhanced AHIP)", RFC-1005, BBN Communications, May 1987

[62] Nagle, J., "Congestion Control in IP/TCP Internetworks", ACM Computer Communications Review, Vol.14, no.4, October 1984.

Index