POSIX Tracing Standard

IEEE POSIX Standard 2001

What is Tracing?

- A way of recording the behavior of an application
  - without having the sources
  - with a minimum of disturbance
  - by tracing both operating system and application generated events

POSIX Tracing Standard

- Known as IEEE Std 1003.1q-2000,
  - work started in 1994
  - official IEEE project (PAR) launched in 1996
- IEEE approved standard, September 2000
- The Open Group Verification Suite available
- Included in the POSIX.1 standard,
  - approved by The Open Group in September 2001
  - approved by IEEE in December 2001
  - soon ISO 9945-2002

Tracing API Inputs

- Real Time OS implementors: Harris, Wind River
- Classic OS implementors: SUN/Solaris, IBM/AIX, Tandem/NonStopKernel
- End Users: DoD
- Universities: Cantabria (Spain)
- Application Developers: Raytheon, DoD

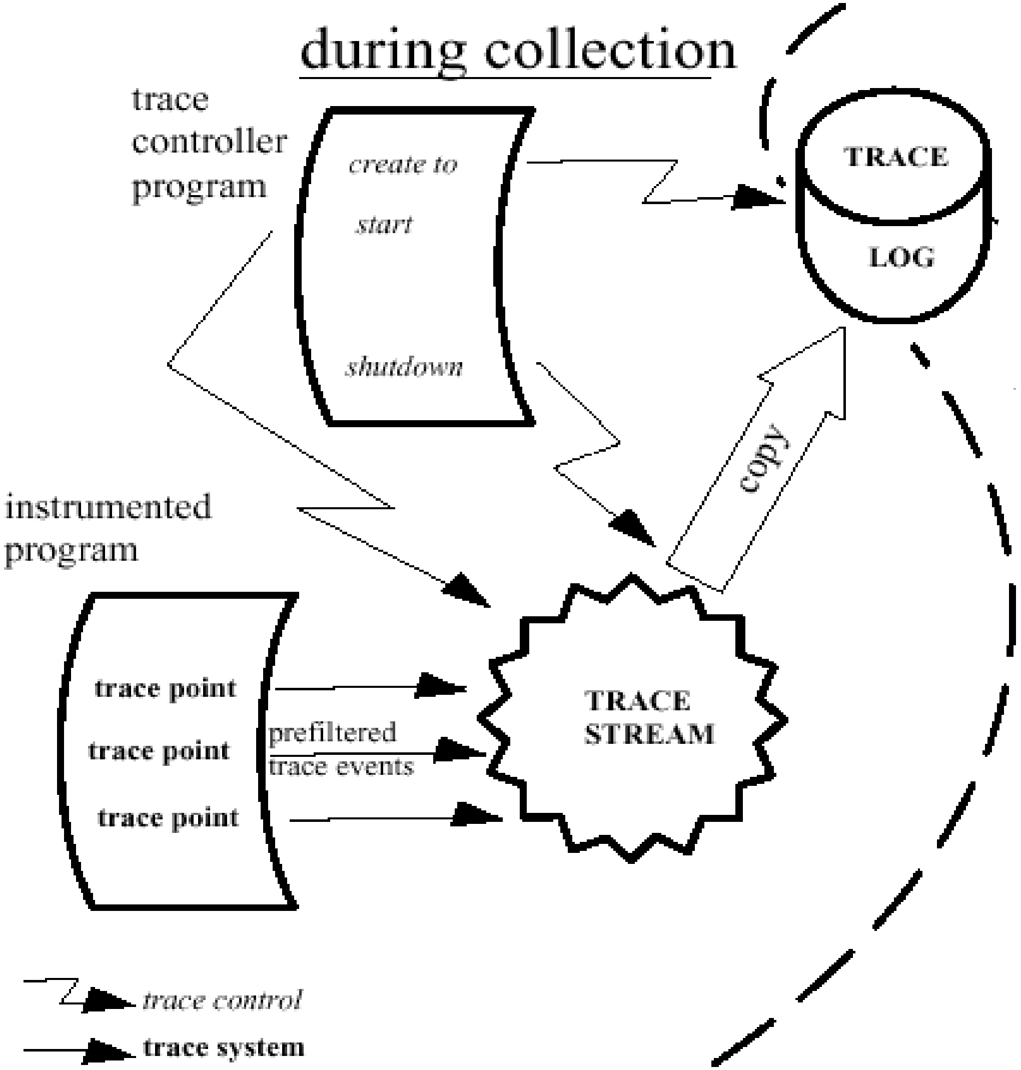
Tracing: State of the Art

- Many OSes already have a tracing subsystem
- Tracing subsystems are not compatible
  - in terms of their APIs
  - in terms of the traced data formats
  - with different levels of abstraction
- Specific trace analysis tool for each OS

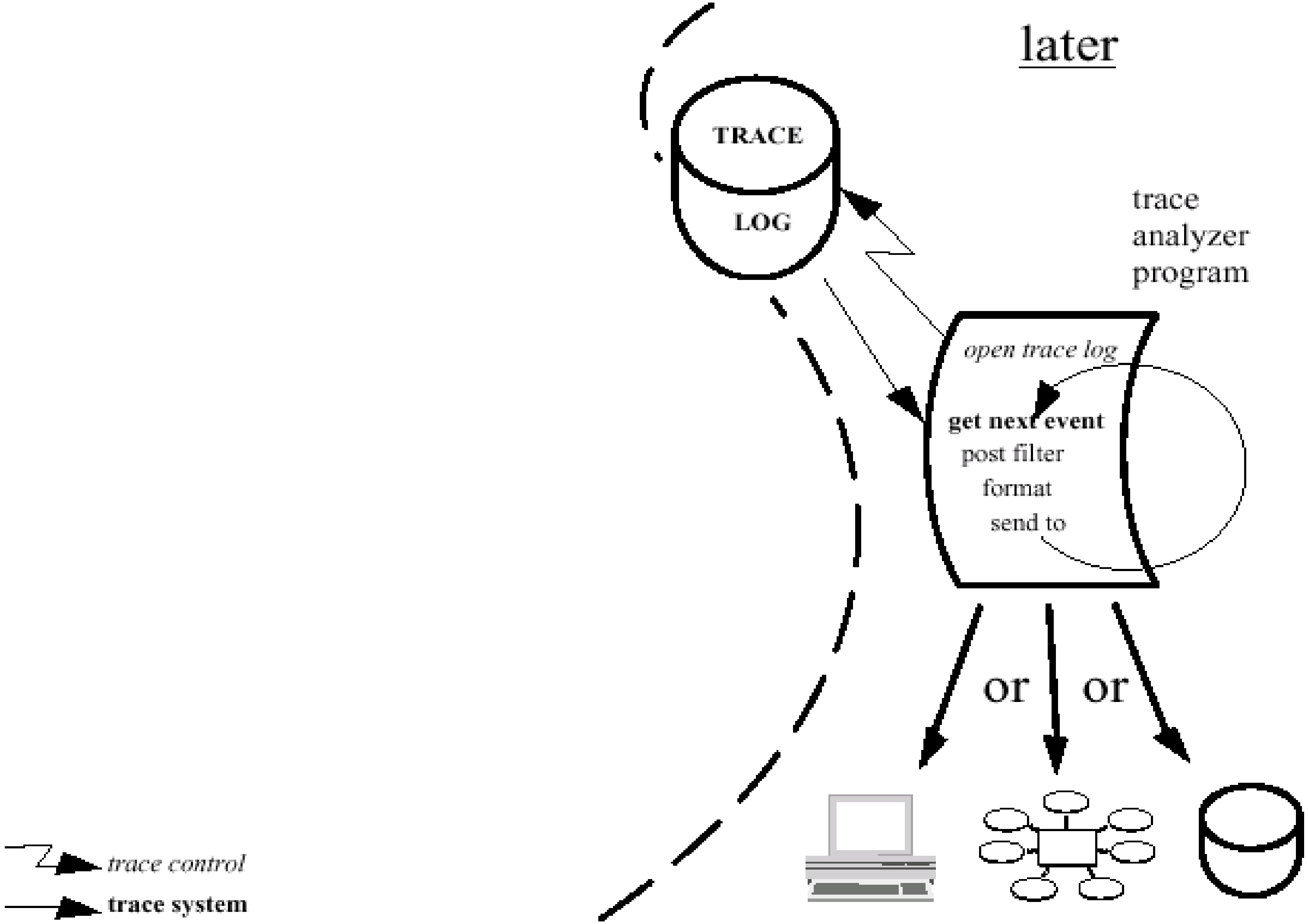
Benefits of a Tracing Standard

- For application providers
  - a common API to instrument applications
- For OS implementors
  - possibility of porting applications increases market base
- For analysis tools providers
  - the same tool for several platforms
- For users
  - a common way to report failure from applications

Tracing Model (1)



Tracing Model (2)



later

TRACE

LOG

trace
analyzer
program

open trace log

get next event

post filter

format

send to

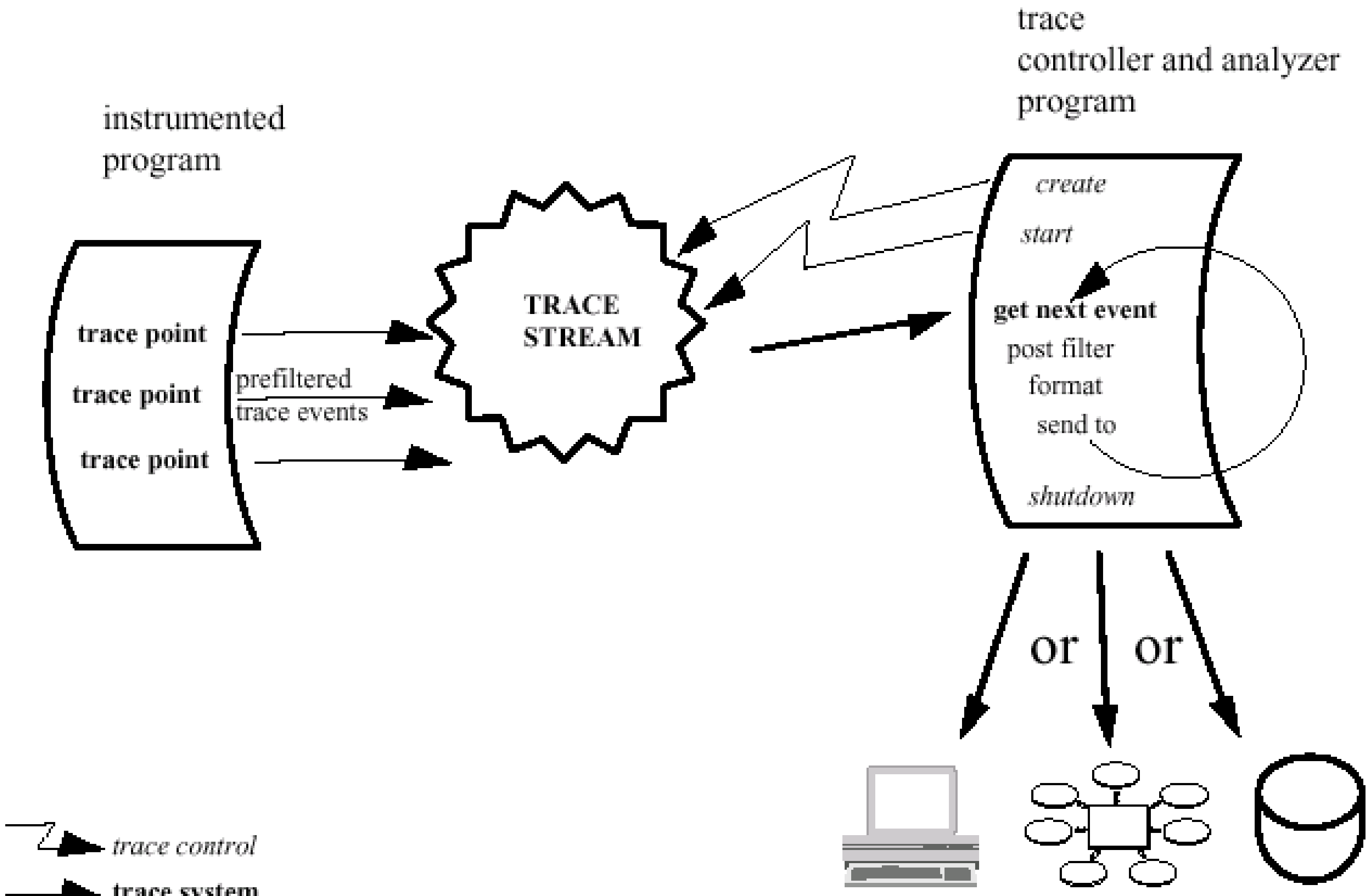
or

or

trace control

trace system

Tracing Model Without Disk



Tracing Roles

- Instrumented-application
  - controllable tracing points
- Tracing Controller
  - monitor trace stream operations from a traced instrumented-application
- Tracing Analyzer
  - get tracing events plus information from a completed trace stream

POSIX Tracing API Features

- Clear tracing API model
- High level of abstraction (opaque data)
  - functions to manipulate opaque data
  - each OS implementation defines its own data format
- Trace event type naming model
- Trace stream storing strategies
- Various information carried by trace event

Tracing Functionality Grouping

- Useful to allow different levels of implementation
- Tracing
  - provides the minimal set of functions for a reduced implementation like an OS without disk
- Event Filtering
  - provides mechanism to select a set of trace events
- Tracing Log
  - provides mechanisms to store a trace stream on disk
- Tracing Inheritance
  - defines inheritance through fork() operation

Final Remarks

- Some Interpretations available
- Already one existing implementation
  - an Ada-language implementation
- For more information, mail to:
  - richef@fr.ibm.com
  - pierre-jean.arcos@windriver.com