

-
-
-

Evolution of an Active Networks Testbed

Steve Berson (ISI), Bob Braden (ISI), Steve Dawson (SRI)

29 May 2002

DARPA Active Networks Conference
and Exposition 2002

(DANCE 2002)

-
-
-

Outline

ABone: A network testbed to support active networks research

- Introduction -- network testbeds & AN node architecture
- ABone Architecture
- ABone Software Components
- Conclusions

-
-
-

Introduction

The ABone has been built and operated by ISI and SRI, with the help and support of many AN research organizations, and with DARPA funding.

- ISI: Bob Berson, Bob Braden, Ted Faber, Siva Jayaraman, Jeff Kann, Craig Milo Rogers, and Yu He.
- SRI: Steve Dawson, Fred Gilham, Marco Molteni, Sonia Tsui, and Arindam Samanta
- Metanetworks: Livio Riciulli

Deconstruct: “*Network testbed to support AN research*”:
Testbed? Active networks research?

-
-
-

Network Testbeds

- A set of shared network resources: nodes and links.
- *Examples of testbed designs:*
 - A. Wide-area overlay testbed: Internet links among nodes at research sites.
 - B. Wide-area testbed: dedicated links* among research sites.
*(*doesn't happen much any more)*
 - C. Cluster testbed: localized set of nodes and links.

-
-
-

Network Testbeds -- Issues

- Testbed design issues:
 - Who owns the testbed nodes, and who controls them?
 - Who has root access?
 - Whether (& how) to virtualize links and processors?
 - Granularity of sharing and isolation:
 - Real node vs. virtual node (vm) vs. process
 - How assemble apparent topology?
 - Hardware switching (e.g., Emulab) vs. software (e.g., XBone).
 - Security

-
-
-

Why a Testbed?

- To share facilities
 - E.g., experiment with 20-100 nodes becomes feasible.
- To extend research into realistic network environments
 - Especially: scale, heterogeneity, robustness.
- To help build research collaborations
- To build a system with common components
- To share tool development and software maintenance overhead.
- To create a teaching platform

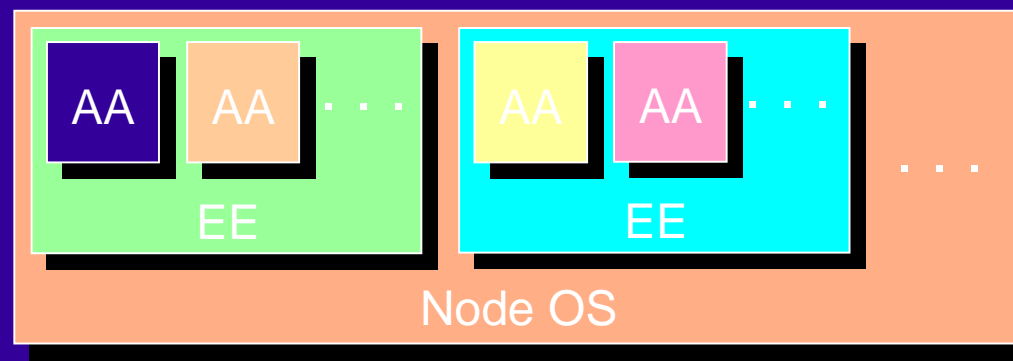
-
-
-

Active Networking ... ?

- Broadly: *Dynamic deployment of programs to process particular packet subflows within the network.*
 - Data plane -- processing data subflows, e.g., adaptive recoding of video.
 - Control plane -- e.g., dynamic installation of flow-specific control/signaling/management algorithms.
- Generally: dynamic deployment of portable code.
 - In-band: capsule (data + program) packets
 - Out of band: dynamic deployment over network.
- DARPA Active Nets research program is exploring these concepts and prototyping technology for it.

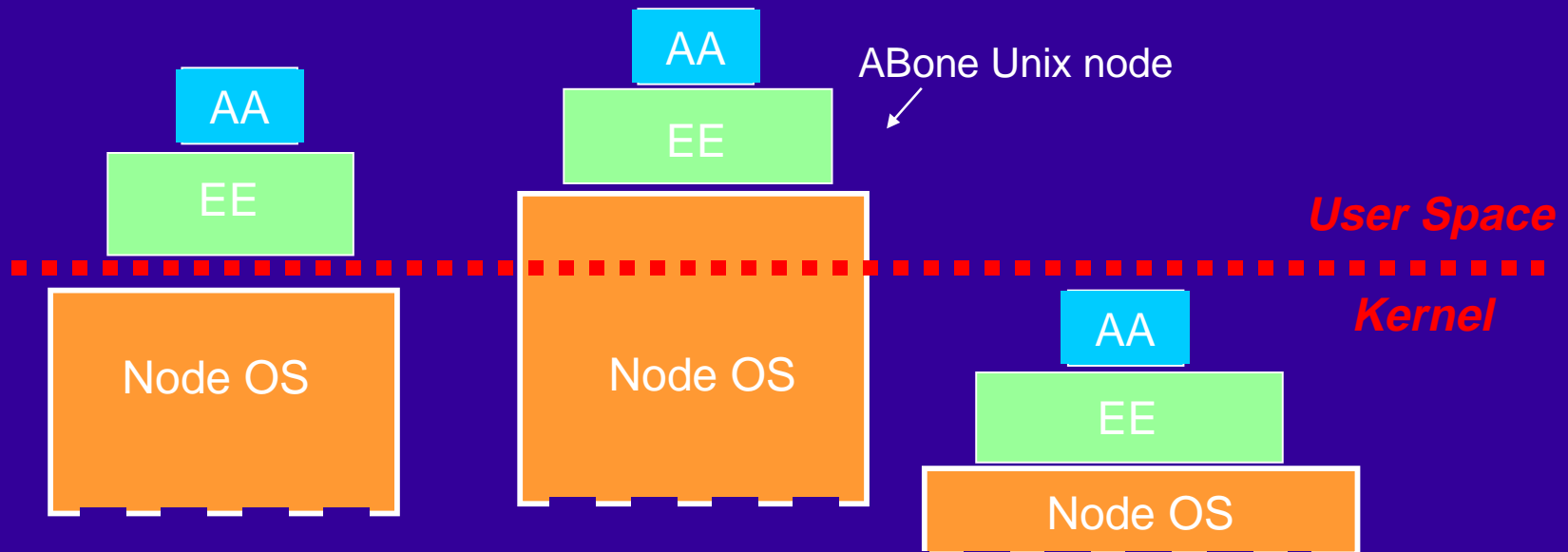
Research Result: Standard Architecture for Active Nodes

- Active Applications (AAs)
 - Fundamental unit of network programming
 - AA code may migrate from node to node
- Execution Environment (EEs)
 - Environment for AA execution (include p-code interpreter)
 - Stable part of software in active node
- Node Operating System (Node OS)



AN Node Architecture

- An EE is installed in a node by/under management control.
- AAs are dynamically deployed and may be transient or persistent.
- Expect: 1 nodeOS, a few EEs, many AAs in each active node.
- Kernel boundary not necessarily at EE/node OS interface.



-
-
-

Some Variations

- Variation 1 [ASP EE, ANTS EE]
 - EE: is effectively a Java-based sub-OS for AA execution.
- Variation 2 [PLAN, SENCOMM]
 - EE: interprets script (carried in an AA) to invoke fixed function library.
- Variation 3 [CANES]
 - EE: Calls plugin modules for generic processing function
 - AA: Set of plugin modules for particular app.

-
-
-

The ABone Architecture

- ABone: (World-)Wide-area testbed for active networks research.
- Nodes: diverse OS platforms distributed across many research organizations.
 - DARPA said: plan for success => 1000 nodes.
 - Actual ABone $O(100)$ nodes.
- Links: Internet overlays, plus dedicated links in DARPA's CAIRN testbed.
- Assumes standard node arch'ture (AA/EE/nodeOS)

-
-
-

ABone Architecture (2)

- Nodes are **locally-administered** by owner sites, who “own” their root passwords.
- AN researchers **remotely install and manage EEs** on these nodes, using ABone software components.
- The security model is a central issue.
 - Central registration of users and nodes.
 - PK security to control who can install what EEs on each node.
 - **Authenticated EE developers are trusted** (&only in user mode)
 - Unix mechanisms in nodes provide isolation and protection.
 - Java-based EEs further sandbox their AAs.

-
-
-

ABone Architecture (3)

- ABone nodes are mostly donated by research sites. AN software must not facilitate break-ins!
 - ORIG: ABone software (Anetd, etc) strictly in **user mode**.
NOW: Netiod, small ABCd subset run as **root**.
 - ORIG: Strictly **production** Unix kernels.
NOW: No kernel mods, but do require optional features (FWs)
FUTURE? Research community build a **modified** kernel for all ABone nodes?

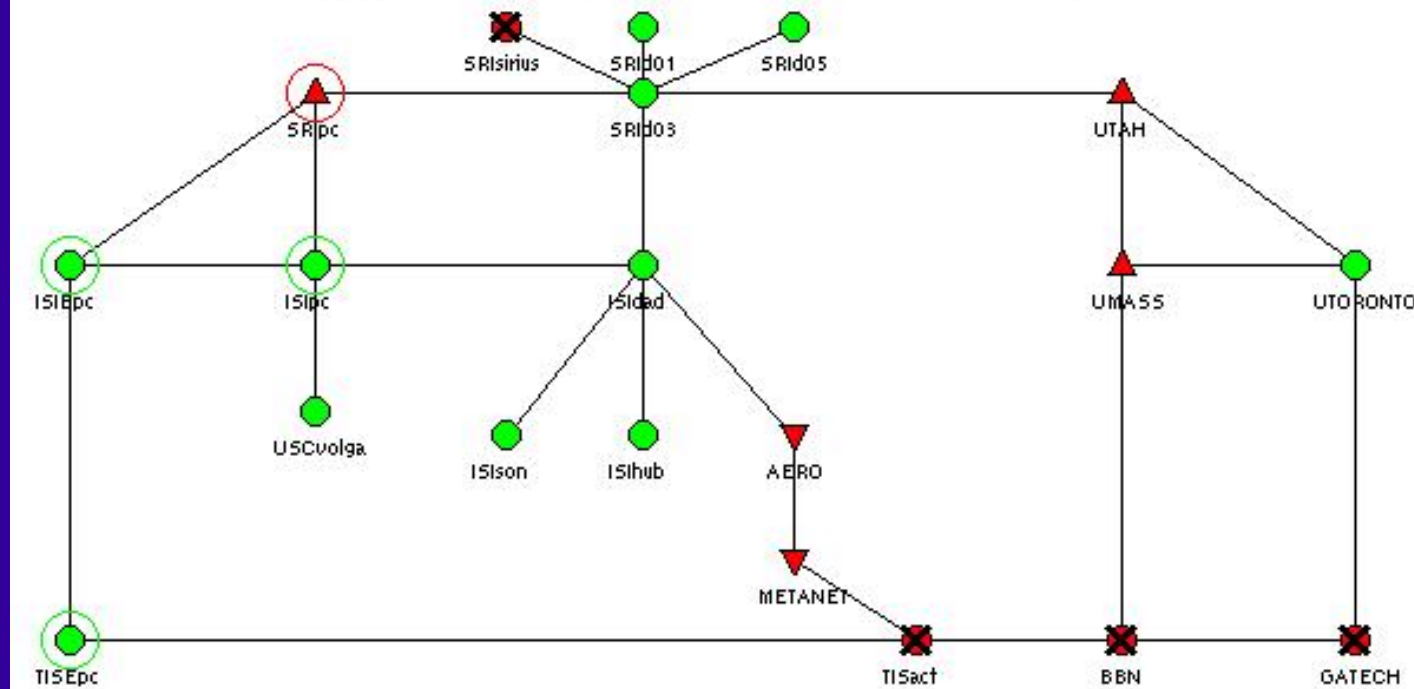
-
-
-

Logical Scope of ABone

- **Unix-based NodeOSs** (Linux, FreeBSD, Solaris)
 - Plus one purpose-built node OS (AMP)
 - No production router platforms or experimental hardware
- **User-level EEs** (Java, C)
 - Node sharing at the process level
 - Cannot push AAs into kernels
- **Permanent EE virtual topologies (overlays)** (ASP, ANTS)
 - Always-available distributed testbed for AA developers.
 - Some support for private EE topologies

Permanent EE Topology

The ASP EE Version 1.4 Public Topology
(ANEP-TypeId #17, UA-API #7889)
Last Updated at 3001128.194403Z (11:44:03 PST)



Monitor display gathered using active probing.

-
-
-

ABone Architecture -- Topologies

- Creating and using a Virtual EE topology--
 - a. Allocate nodes
 - b. Build/allocate accounts on these nodes
 - c. Define desired topology
 - d. Generate & install configuration files on nodes
 - e. Start the EEs on nodes
 - f. Monitor topology
 - g. Launch an AA to run the experiment
- **Permanent EE topologies:** ABOCC does a-f statically. For **private topologies**, client tools can do a-f more or less automagically.

-
-
-

Varied Network I/O Modes

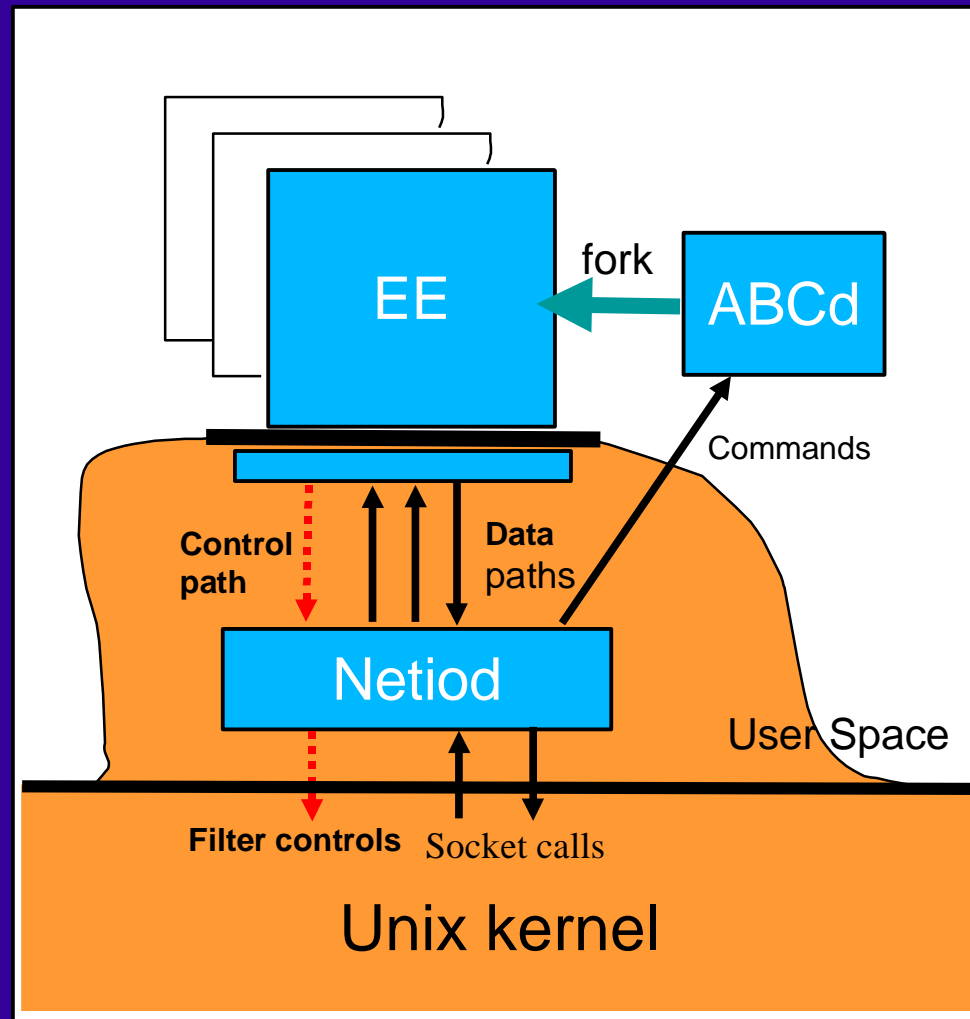
- **Virtual connectivity (UDP overlays)** (ANTS & ASP EEs)
 - Per-EE virtual topology & virtual network addr. space.
- **Native IP connectivity** (ASP EEs)
 - Running in the Internet ‘porridge’ with real IP addresses.
 - Depend upon physical topology
- **Link-Layer connectivity** (ASP EEs)
- **Virtual native IP connectivity** (Xbone)

-
-
-

ABone Software Components

- On each ABone node:
 - ABCd (Anetd): Remote EE management daemon
 - Load and launch an EE (Java or C) in a specific file subspace
 - Terminate, restart, configure, and monitor EE.
 - Netiod: Network I/O daemon
 - Runs as root for kernel filtering
 - Provides uniform interface across Unix platforms.
- At client site:
 - ABCd (Anetd) client and ABoneShell interface.
- At central site:
 - Web-based registry program.

ABone Software Components (2)



-
-
-

ABOCC -- ABone Coord'n Center

The “operational” side of the ABone -- the ABOCC:

- Installs and updates ABone software components.
- Maintains a web-based registry of nodes and users.
- Maintains and distributes access control files.
- Installs and monitors permanent EE topologies.
- Coordinates problem reports and fixes.
- Supports users.

ABone Nodes

Registered ABone Nodes

Suspend	Reload	SaveCoord	Table
Unknown status			
No Response			
ICMP Ping OK, but no ANetD response			
ANetD is running but denying access			
ANetD is running and responding correctly			

US: 82 (78 DARPA,
14 CAIRN)

France: 4

Canada: 3

Italy: 2

Australia, Finland,
Germany, South
Korea, Taiwan:
1 each.

96 registered nodes

-
-
-

See DANCE 2002 Paper for...

- Precisely how ABCd (and Anetd) work
- Core vs. edge nodes and the DANTE protocol
- The web-based registry function
- The complete ABone security model & mechanism
- File system and account configurations
- ABoneShell and EElets for management
- Using the NodeOS channel abstraction for network I/O
- Active Networks security implementation in the ABone
- Distributed debugging

-
-
-

Partial List of ABone Users

- Aerospace/TASC/UMass. -- Active Filtering: SANDS
- BBN Sencomm Project -- Network Management
- Columbia University -- NESTOR
- NAI -- AMP nodeOS
- Ga Tech -- CANES EE
- ISI -- ASP EE
- TASC/UMass: AER/NCA Reliable Multicasting
- Univ Kentucky -- Concast
- Utah/U Washington -- ANTS EE
- USC -- grad student projects
- ...

-
-
-

Conclusions

- We built an AN testbed, the ABone, but ... *it is just beginning to play a significant role, at the end of the AN research program.*
 - We emphasized AA development over EE development.
 - Some of the EE prototypes do not map easily into a wide-area testbed.
- Lessons:
 - To be effective, a testbed needs to be part of the research program from the beginning.
 - Both wide-area and cluster testbeds are needed, and they should be integrated.

-
-
-

AN Testbed Issues

- To what extent can EEs be portable across ABone platforms -- Unix and specialized Node OSs?
- Should we accept the limitations of production Unix kernels for the testbed, or use modified kernels?
 - ABOCC could supply kernels for all nodes -- significantly reducing heterogeneity, at the cost of some currency.
 - Research community could build common platform.
- [How] can we (safely) download AAs into the kernels of testbed nodes?
- Effective integration of wide-area and cluster testbeds

-
-
-

AN Programming Language

- Java was the obvious choice for active net programming environment, but was it the right choice?
 - Java has been fairly portable across platforms, but not across versions.

Java has been a moving target, seriously increasing its cost. The product space of Linux, FreeBSD versions with JVM versions is huge and full of potholes.
 - Using Java for systems programming has exercised its least stable and buggiest features, especially threading & security.
 - The active nets program needs more stable implementations and perhaps a better language.

-
-
-

Active Networking: Parting Shots

- It is time to publish our major documents as RFCs
- *There IS a pony in here somewhere...*
 - We can and should build a unified technology base for applying active networking to real and important networking problems.
 - (Some of my most end-to-end friends, once severe critics, are ready to be convinced now)
 - Killer apps: middlebox control, intrusion response, network management, signaling, protocol experimentation.
 - Is there currently a funding source with the resources and culture to realize this potential?