Active Networks

IEEE Infocom 1999 Tutorial
March 22nd, 1999

Jonathan Smith

University of Pennsylvania

http://www.cis.upenn.edu/~jms

Acknowledgments:

- All Penn work and most other work supported by DARPA ITO.
- Collaborators: Alexander, Arbaugh, Farber, Feldmeier, Gunter, Hadzic, Keromytis, Marcus, McAuley, Menage, Nettles, Segal and Sincoskie...
- Hewlett-Packard, Intel and 3Com

Tutorial Outline:

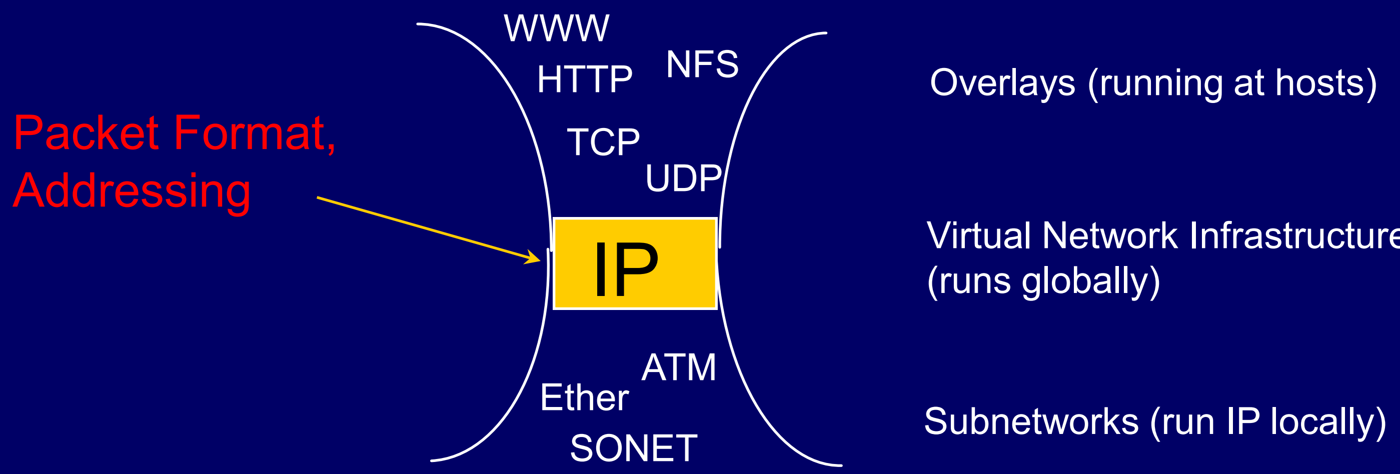
- Introduction and Background
- Architecture for Active Networks
  - Execution Environments (EEs)
  - Node Operating System (NodeOS)
  - Security Architecture
- Applications
- Interoperability
- The Future

1. Introduction

- The Traditional Model: Store & Forward
- Why do we need Active Networks?
- New Model: Store, *Compute* & Forward
- Roles for Languages, O.S. and Algs.
- Nodes, Protocols and Networks

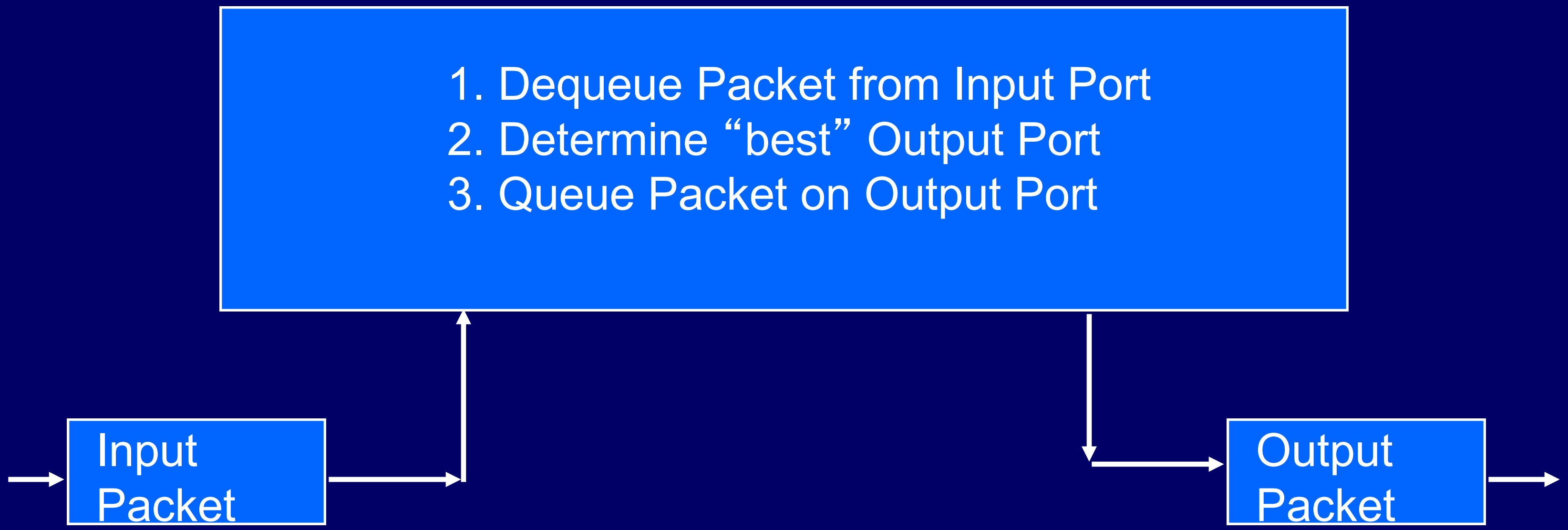
Virtual Infrastructures, e.g., IP

- IP is a network interoperability layer
- Interoperable through minimality:



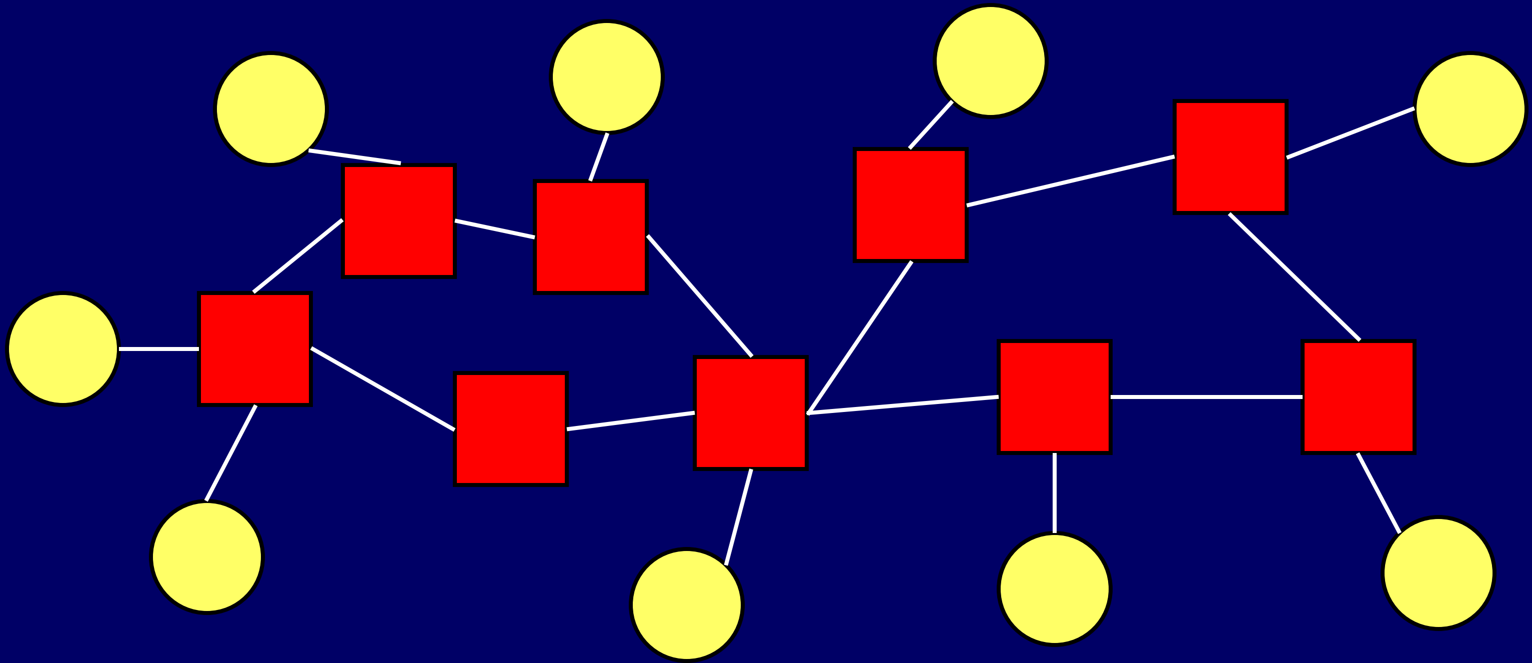
IP Routing Infrastructure

- Model: Store and Forward



"Passive" Networking

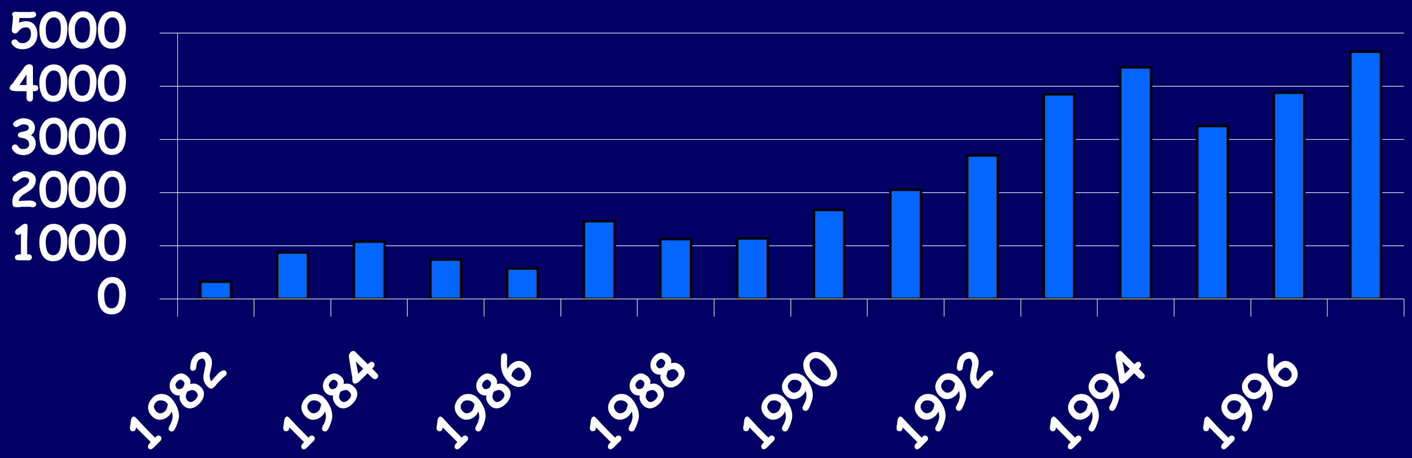
- Smart hosts on the edges
- Passive switches in the center



Challenges as Internet scales

Hope	Reality
"Adding New Protocols is EASY"	RFC 1112, 1989 Ubiquitous Multicast, 2009? ☺
"End to End QoS"	ISSSL…
"All Intelligence at the Hosts"	Not source routed
"IP can run over two cans and a string"	And TCP can't figure out if the string is congested or fraying…

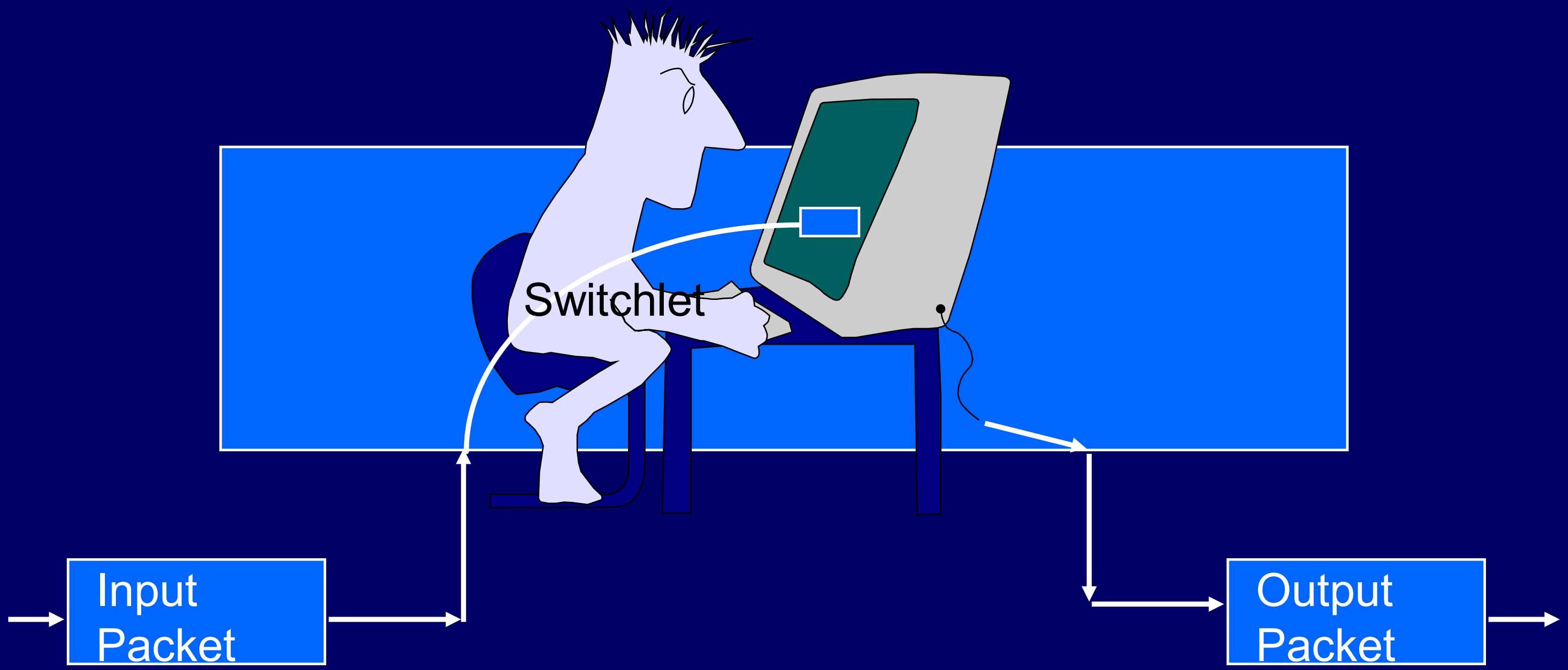
RFC Pages by Year 1982-1997



And 13797 pages draft in ‘97 (9/16/97)….

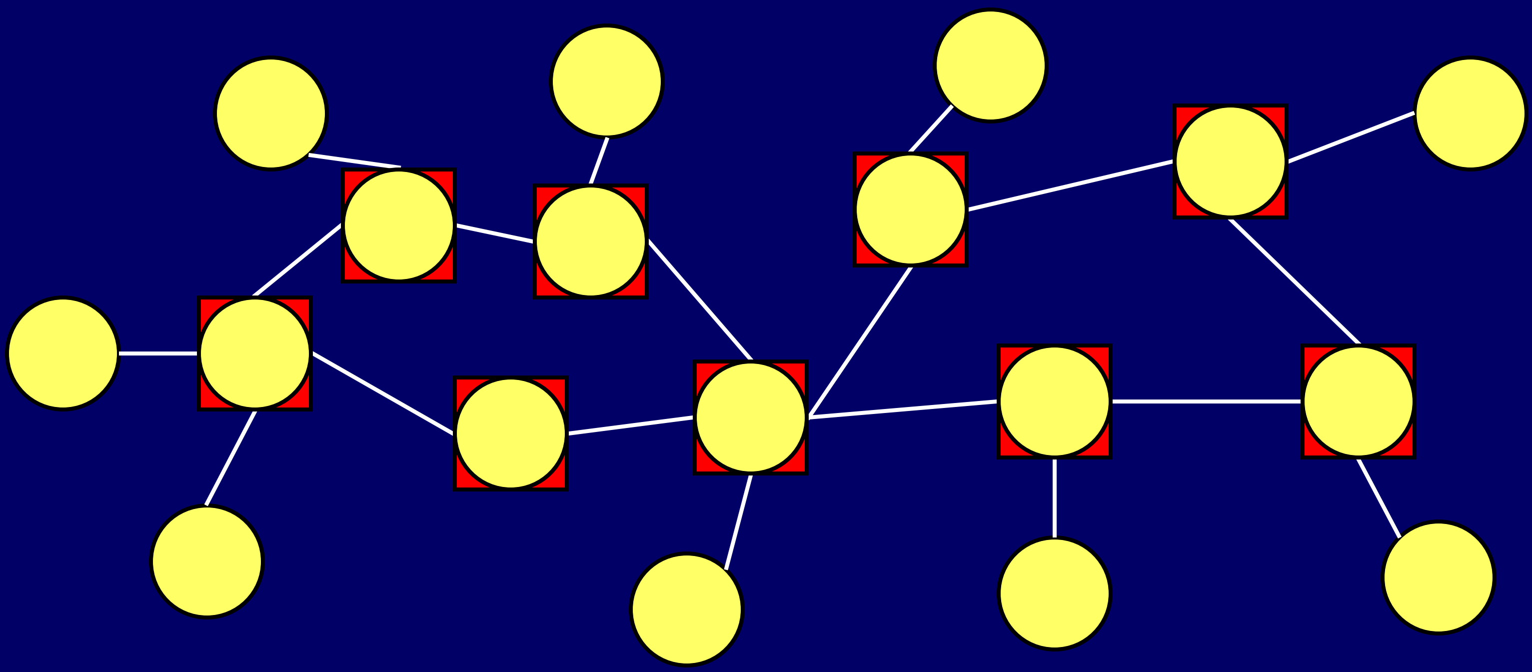
Active Networking Nodes

- Store, COMPUTE and Forward!



Active Network Model

- Packets ("switchlets") can change the behavior of the switches "on-the-fly"
  - In-band active packets
  - Out-of-band active extensions



Result: ‘‘Active’’ Networks

- Accelerate service creation with programmable network infrastructure
- Programmable on per-user or per-packet basis
- Is this just another O.S. problem?
- See Tennenhouse, Smith, et al. survey in IEEE Network Magazine, Jan. 1997

Why Do This?

- Faster response to problems and possibilities in network
- Per-user protocols
- Allows experimentation
- Accelerates network evolution
- Examples
  - Web proxy caching
  - Auctions
  - Reliable multicast
  - Congestion control

Activations (Less radical)

- Network control and measurement
  - flow and congestion control (self payment)
  - measurement (at intermediate nodes)
  - real-time self-expiring packets
- Break “invisibility” of data link layer!
- Sanity, e.g., Anti-spam filter
- Inverse multicasting

Activations (More Radical):

- Self-Paying Information Transport
  - Routing by economics; policy with $$$
- Use Multiple Routes other than failover
  - Route BONDing (striping)
  - Diversity routing
  - Routing is not infrastructure!
- Sensor fusion, DNS & WWW caches, etc.
  - Address latency with *Architecture*

Questions for Active Networking

- How to do it
- Where (not whether) to do it:
  - under IP
  - IP
  - over IP
  - management plane, applications, etc.
- Can it be deployed?

2. What's known?

- Experience with user programmability
- Process Migration
- SPIN and Exokernel
- SOFTNET

User Programmability

- EMACS uses LISP for extensibility
- MACRO packages pervasive
  - LaTeX, troff, MS-Word
- Metamail - LISP extensions
- PostScript printers

Process Migration

- Move running code from place to place
  - load balancing / large data
- Investigated heavily in late 1980s
- Machine heterogeneity and state major challenges
- These challenges were addressed by mobile code systems and languages such as Java.

The U. Washington SPIN O.S.

- Dynamically modifiable O.S.
- Programs loaded in "on-the-fly"
- Programmed in Modula-3
- High performance, sandbox-style security, safety from language
- Many lessons for active networking

The MIT Exokernel

- User-specialized operating system
- Minimal required resources
- Other elements implemented as privileged or unprivileged libraries
- Many networking applications
  - DPF
  - PAN

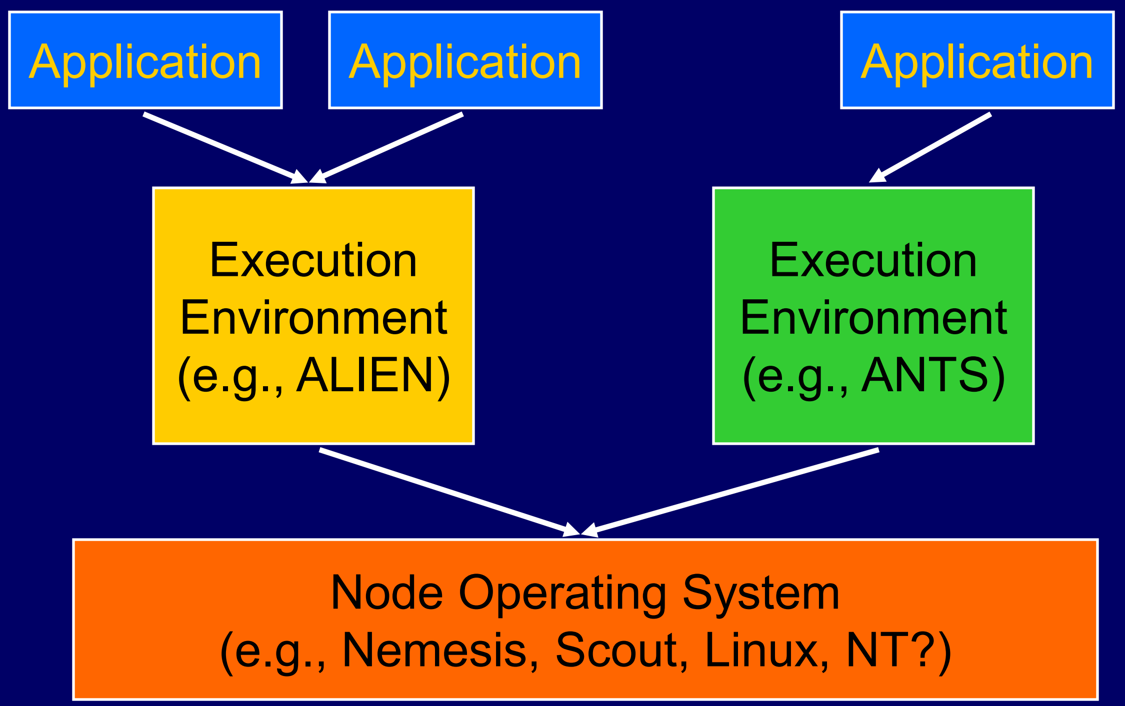
U. Linkoping SOFTNET system

- Mid-1980s (thus really first A.N.)
- Radio nodes with processors (6502s) and multi-tasking Forth operating system
- Demonstrated operational dynamic network architecture - Forth loaded on the fly
- Lessons in architecture and security

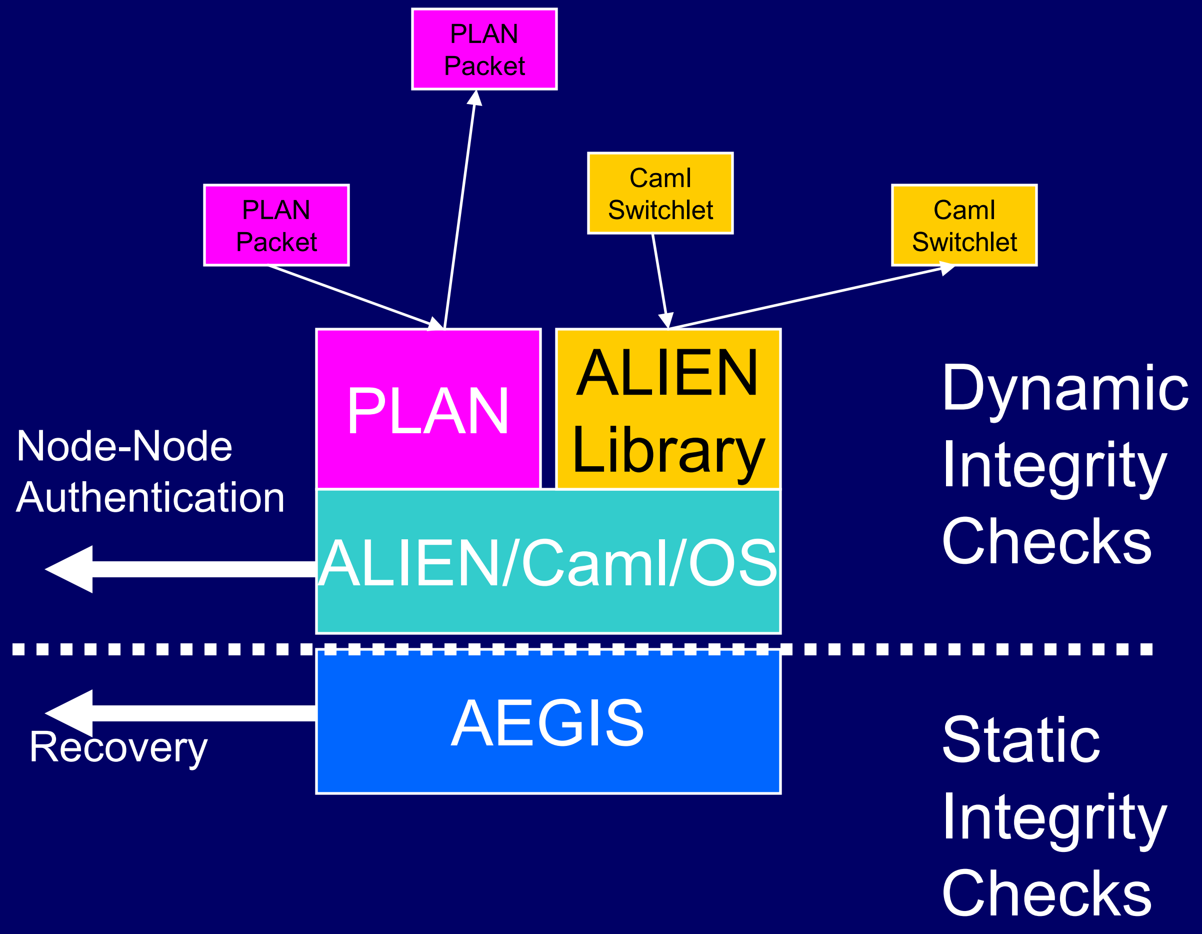
3. Active Net Architecture

- User Programs
- Execution Environments (EEs)
- Node Operating System (NodeOS)
- Security Architecture

“Active Network Architecture”



Example: SwitchWare Architecture



4. Execution Environments

- BBN Smartpackets
- MIT ANTS
- Penn PLAN
- Columbia Netscript
- Penn ALIEN
  - Detailed Case Study

The Design Space

- Usability *vs.* Flexibility *vs.* Security *vs.* Performance
- A General-Purpose Language gets the first two for free; other two are <u>hard</u>!
- Domain-specific Languages may achieve different tradeoffs

Programming Language Features

- Strong typing
- Garbage collection
- Module thinning
- Dynamic loading
- Platform independent representation of switchlets
- Performance
- Also desire threads and static typing

Caml and Java: Features

	Caml	Java
Dynamic Loading	X	X
GC	X	X
strong typing	X	X
array bounds	X	X
static typing	X	?
compact format	X	X
arch independent	X	X
reasoning	X	
low level access	X	

BBN Smart Packets

- Domain-specific language
  - Source code - Sprocket
  - Comiples to stack-based CISC - Spanner
- Designed for network management
  - Spanner runtime accesses MIBs
- Spanner is a compact representation
  - fits in an Ethernet frame

MIT Active Network Transport System (ANTS)

- Wetherall, et al., OpenArch '97
- Largely a library of Java
- Uses the Java JVM runtime
- Packets carry 1 function each
- Packets "drag" code after them into a soft-state-like function cache so that subsequent packets run fast

Packet Language for Active Networks (PLAN)

- Hicks, Kakkar, Moore, Gunter, Nettles
- Capsule-based approach
- CAML runtime
- Highly-restricted domain specific language (a safe "glue" language, like the UNIX shell), extensible via ALIEN
- Active extensions do restricted things

Netscript

- Yemini and daSilva at Columbia
- Domain-specific
  - Network Management
  - Virtual network configuration
- Implemented in Java
- Creates dataflow mesh
- Used for dynamic firewall configuration

The ALIEN Active Loader

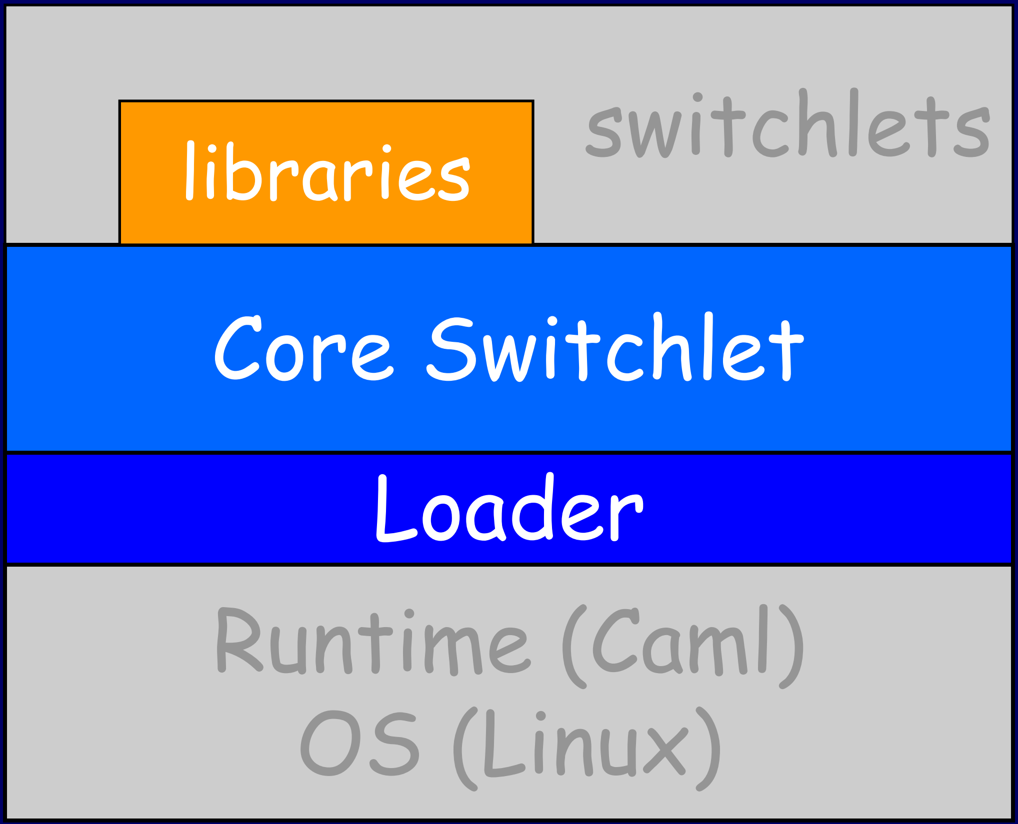
- D. Scott Alexander
- CAML runtime
- CAML capsules restricted via module thinning
- Digitally-signed certificates for remote accesses to resources
- Will use for detailed case study

The ALIEN Approach

- Achieved by *restricting* a general computing model
- Realized in ALIEN, an active loader for Caml
  - General computing model
  - Interface to OS
  - Interface to active code
- Only privileged portions of the system can directly access shared resources

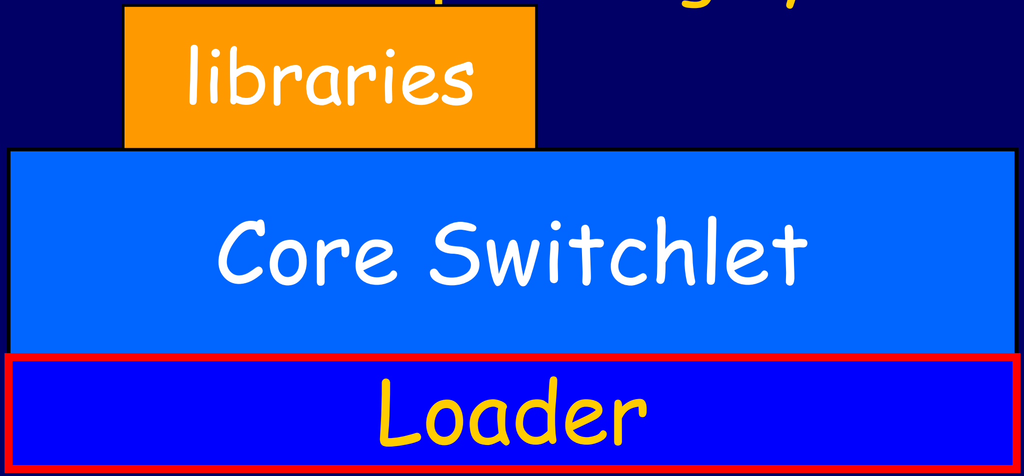
ALIEN in an Active Element

- Three layer architecture



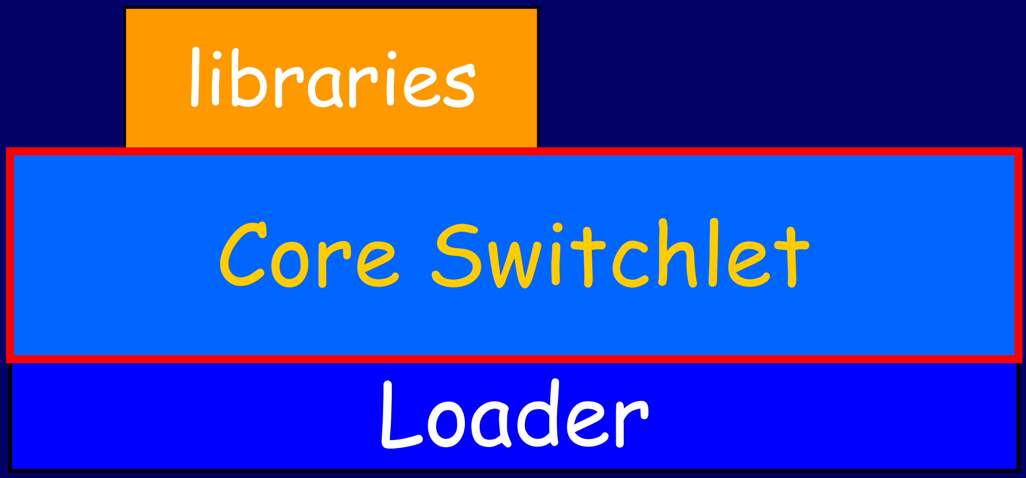
ALIEN Loader

- As small as possible
- Fixed (but overlays can mask)
- Mechanism rather than policy
- Basis of privilege
- Interface to operating system



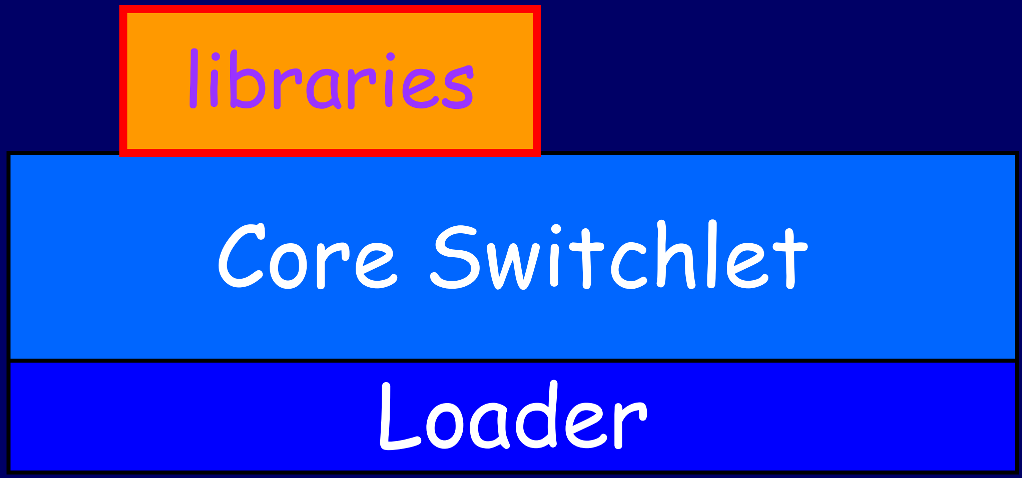
The Core Switchlet

- Policy *and* mechanism
- Privileged
- Loadable (conceptually, Caml does not allow)
- Interface to switchlets



ALIEN Libraries

- Less well-defined because of ease of change
- IP, UDP, crypto routines, checksums
- Unprivileged



Locating Functionality

- Prefer libraries
- Isolate privileged functionality
- Policy is managed in Core Switchlet
- Expand Loader only to allow Core Switchlet startup

Implementation of Switchlets

- Active Extensions
  - Loaded from disk or network (TFTP)
  - We use queues for communication
  - Could use upcalls...
    - Security?
  - ...or blocking downcalls
- Active Packets
  - ANEP encapsulated (over UDP or link layer)
  - Can use SANE for security
  - Linker/ procedure call for communications

Active Packets in ALIEN

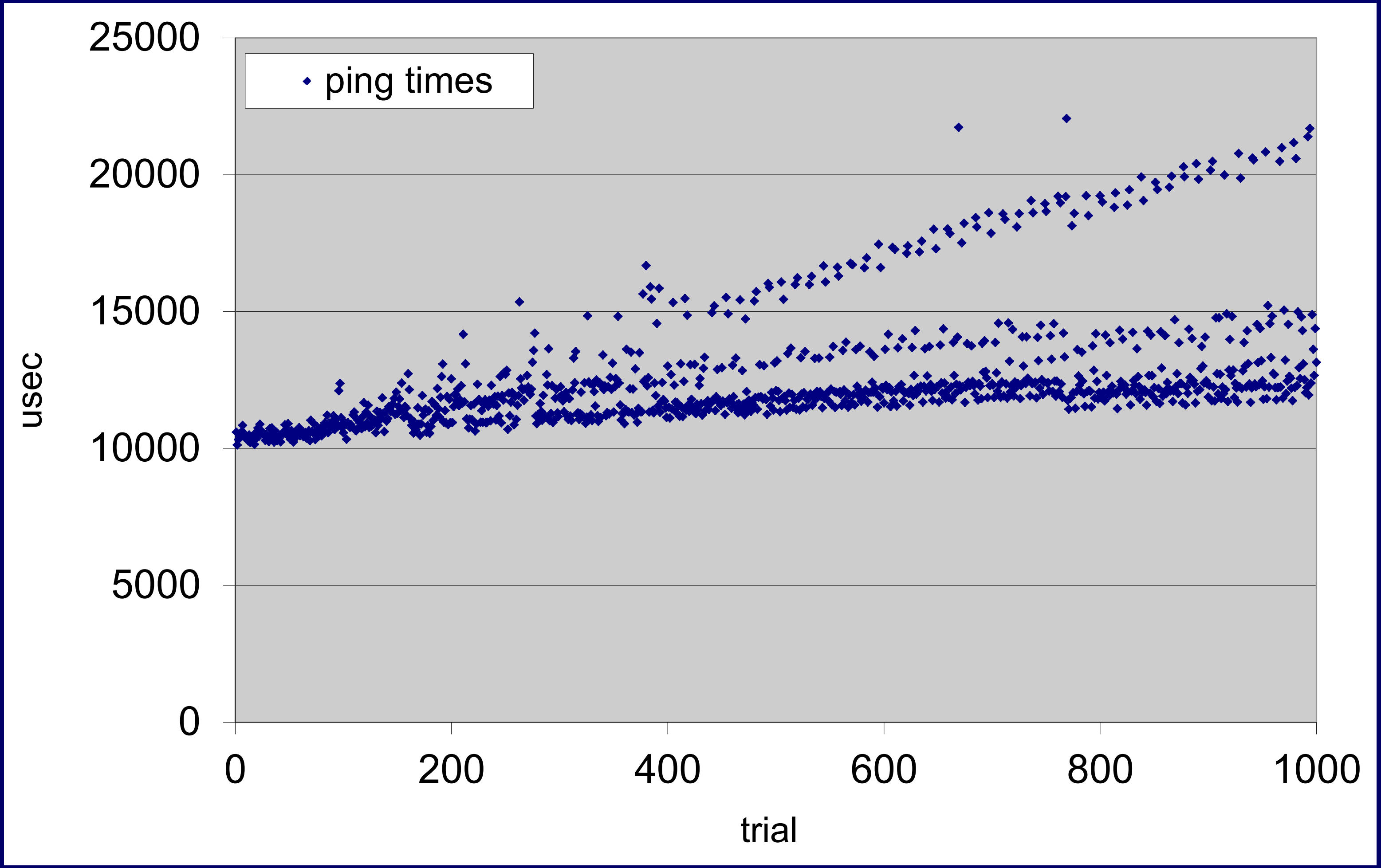
- If ANEP header indicates ALIEN
  - SANE processing as part of ANEP
  - Code portion is loaded
  - *func* is called with code, data, and func name as arguments

link layer header	ANEP header/ SANE auth	code portion	data portion	func name

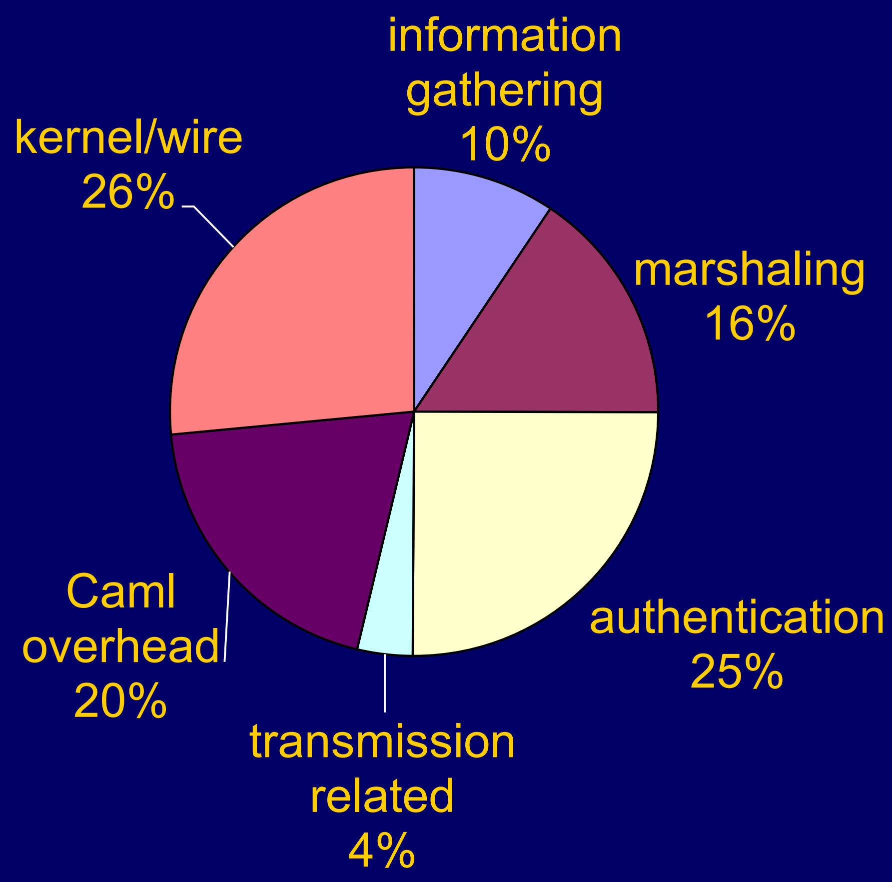
Experimental Setup

- DEC Alpha 21164SX @ 533Mhz
- 128 MB ECC synchronous DRAM
- L1 cache: 16KB instruction; 8KB data
- L2 cache: 1MB synchronous pipeline burst
- 100Mb Ethernet: DEC DS21140 "Tulip"
  - Full-duplex, cross-over cable
- Timing via rpcc instruction (cycle counter)
  - Cost 2 cycles to time C code
  - Cost 1.5 - 2 μsec for Caml code

saneping Performance



Overall Breakdown of Costs



Major Costs

- Kernel/Wire (26%, 3078 μs)
  - Kernel time + transmission time
  - To avoid
    - Reduce size of packet
    - Reduce or avoid kernel boundary crossing cost
- Authentication (25%, 2910 μs)
  - Mostly cost of performing SHA-1 (4 times)

Controllable Costs

- Caml overhead (41%, 2296 μs)
  - Cost to link byte code file into ALIEN
- Marshaling (32%, 1816 μs)
  - Difficulty comes from maintaining type safety
  - Could improve with limited version of Marshal
  - Could improve further with SANE?
    - Sending hosts signs any valid object not created with limited version of Marshal
    - Cost? Benefit?

Controllable Costs (con't)

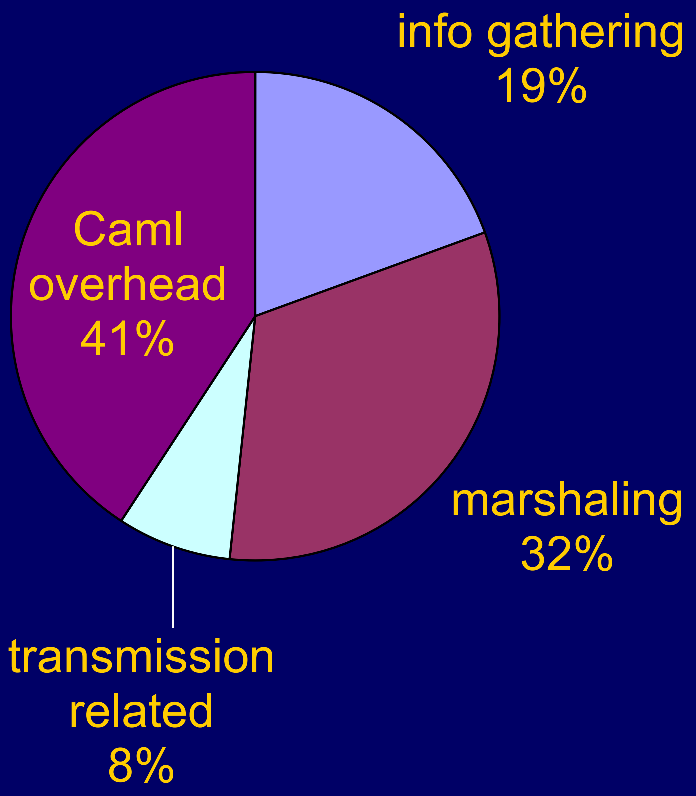
- Information gathering (19%, 1094 μs)
  - Finding a route, finding an addr, reading byte code file
  - Cost of reading byte code is 1043 μs
- Transmission related costs (8%, 425 μs)
  - Call to Udp.sendto_udp
  - Call to queue up packet for receiver
  - Might be reduced by upcalls

Cryptography is Expensive

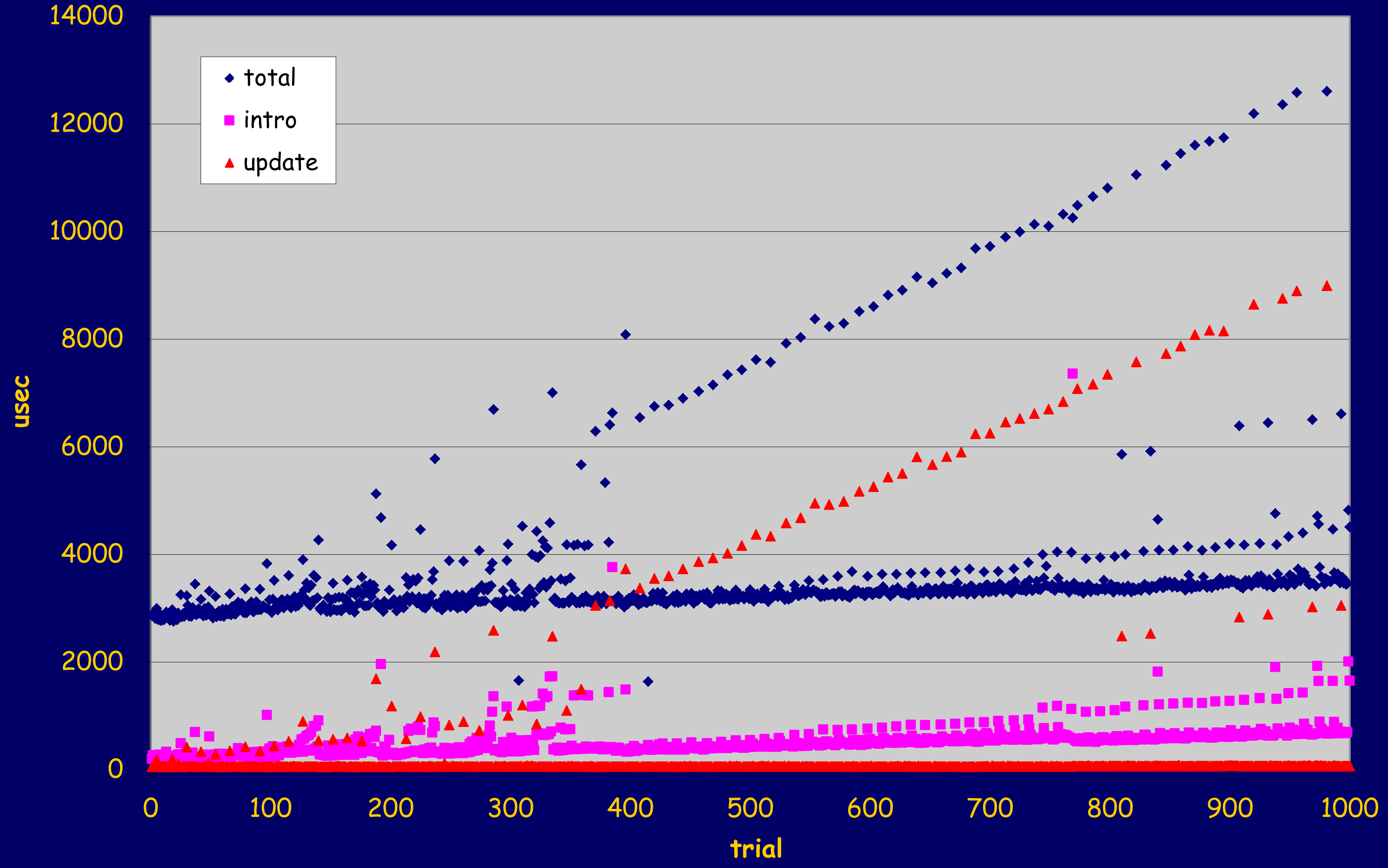
- Implemented in C because too slow in Caml
- Times to hash 4MB of data

	bytecode	native
Caml Int32	86.45s	61.99s
Caml int	36.03s	2.48s
C		0.33s

Breakdown of Caml Costs



Caml Overheads



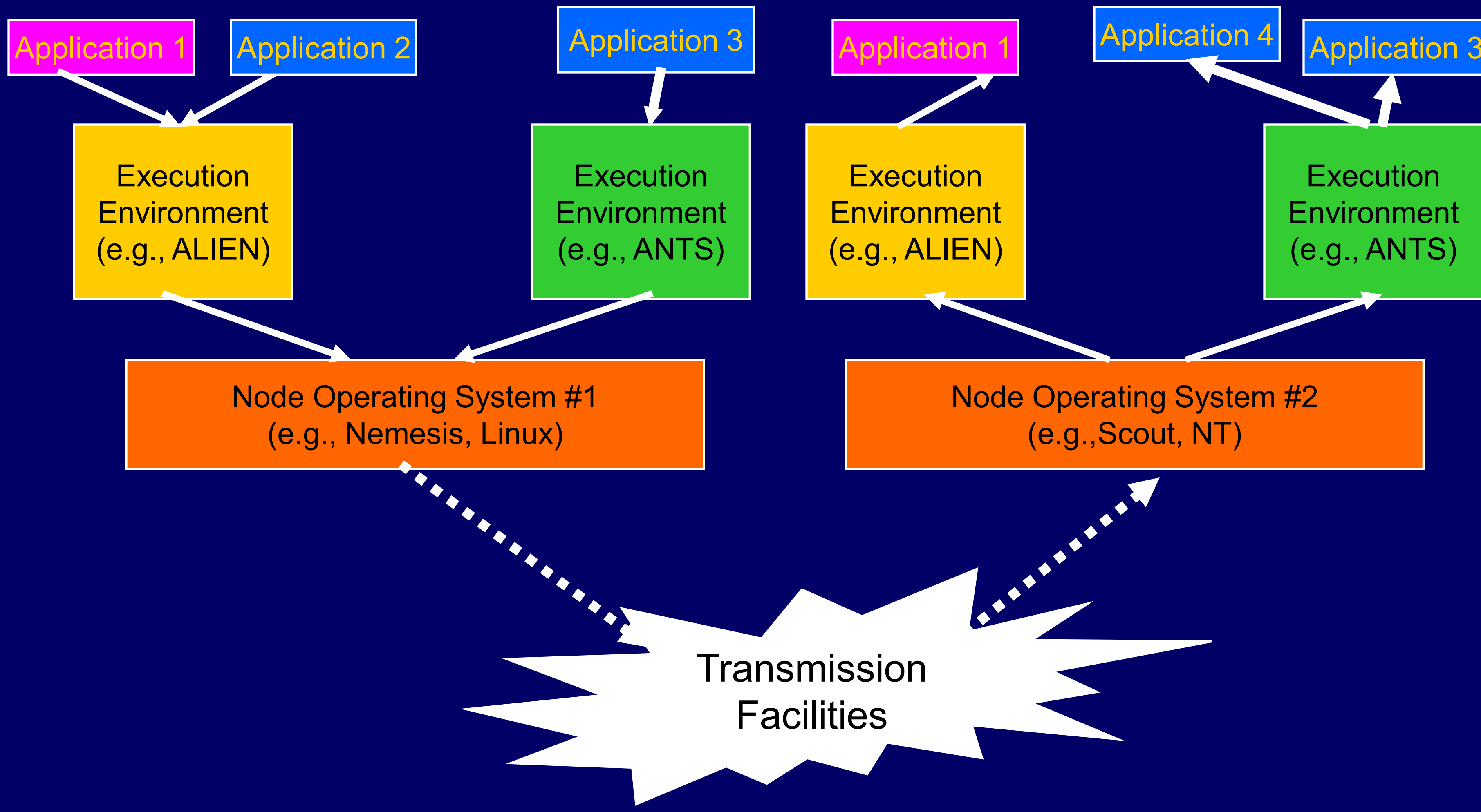
Caml Overheads (con't)

- Reallocate global_data every 12 - 13 pings
- Cost of "intro" rises due to increase in symbol table size
- Jumps in "intro" due to hash table resizing

Other Runtime Issues

- Transmitted ping program is 2454 bytes
- Loading a module from Linux buffer cache costs 3ms
  - We added an extension to load from memory
- Scheduler costs between 100μs and 250μs
  - If any thread desires I/O select() is called
  - We structure our code to avoid (some) calls to the scheduler to get these results
  - We will need to use a different scheduler

Internode Interoperation



EE Deployment Challenges

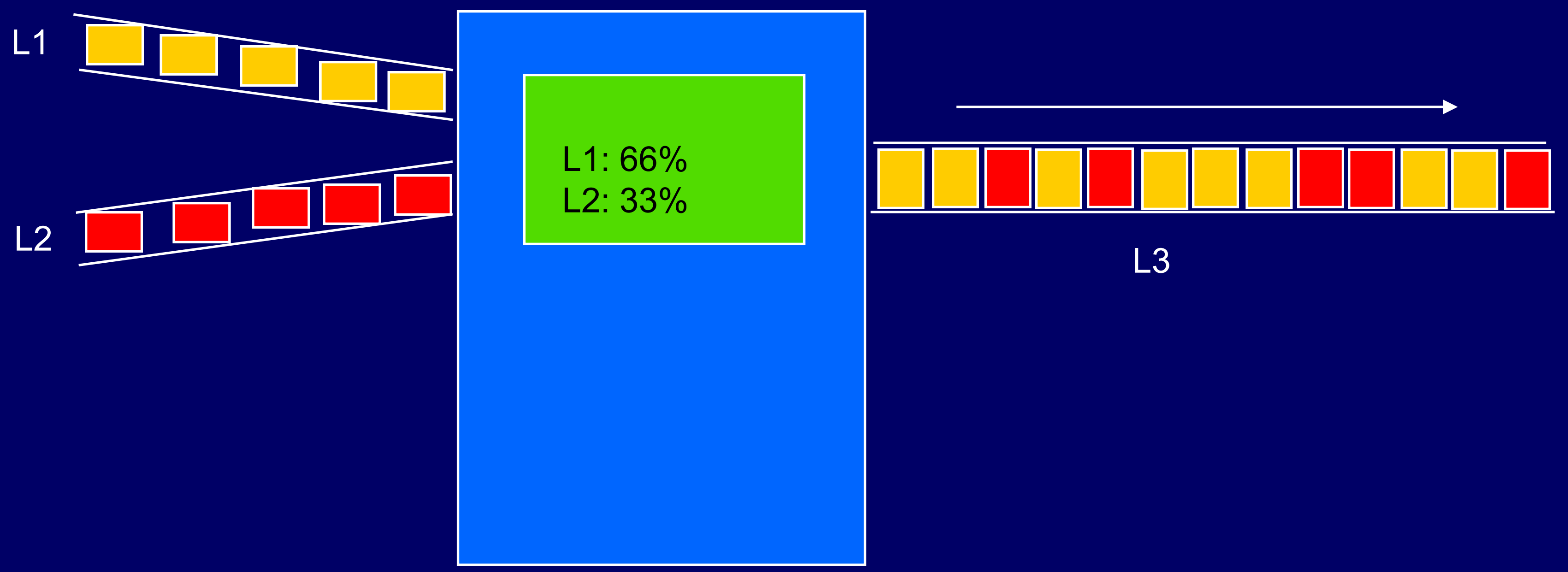
- EE interoperability
  - Will we need an EE-interoperability EE?
  - Or will we be limited to a subset of nodes?
  - Difficulties with P.L.-based security
- Local Autonomy vs. Global behavior
- Varying capabilities of NodeOS?

5. Node Operating System

- Aggregation and Flows
- Resource Management and QoS

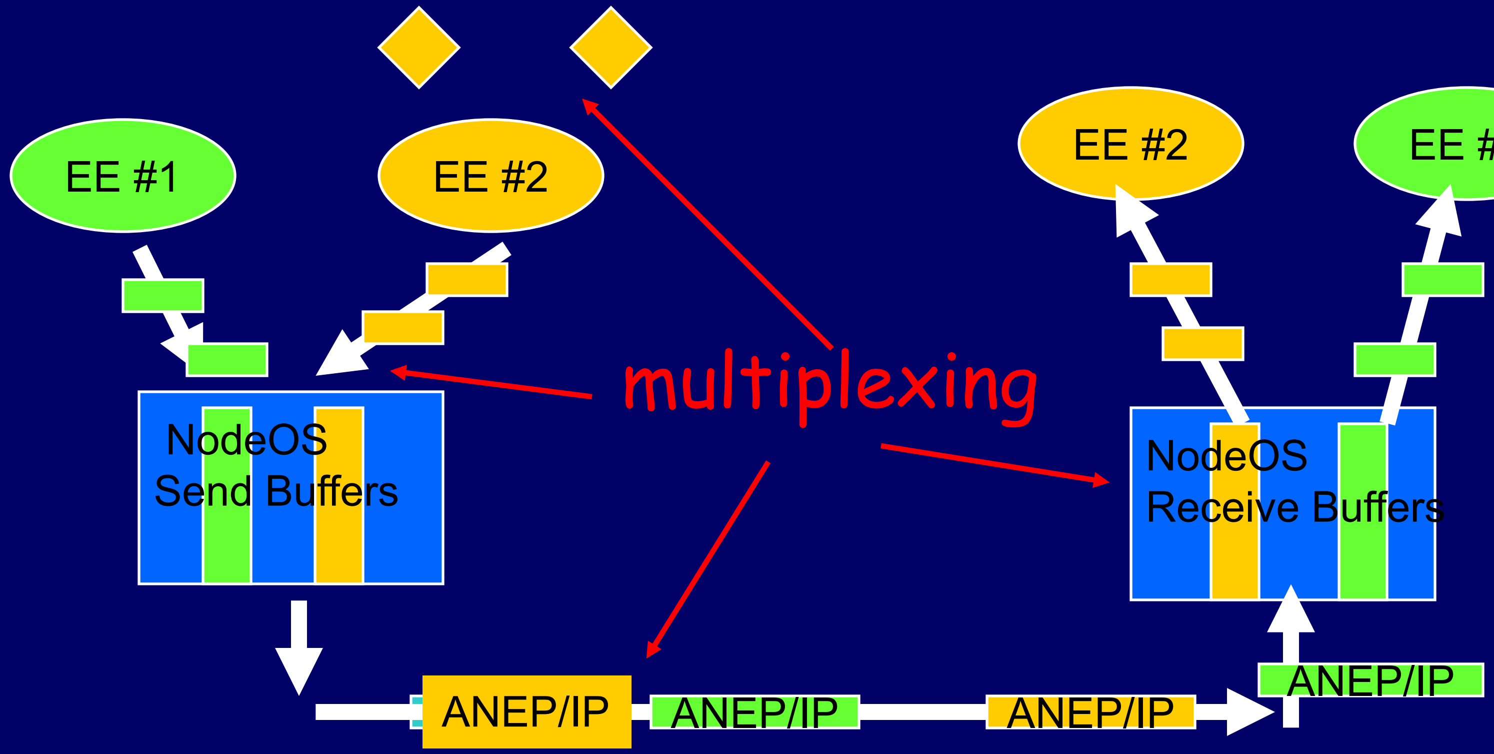
Need to control multiplexing

- E.g., assign L3 bandwidth 66%/33%



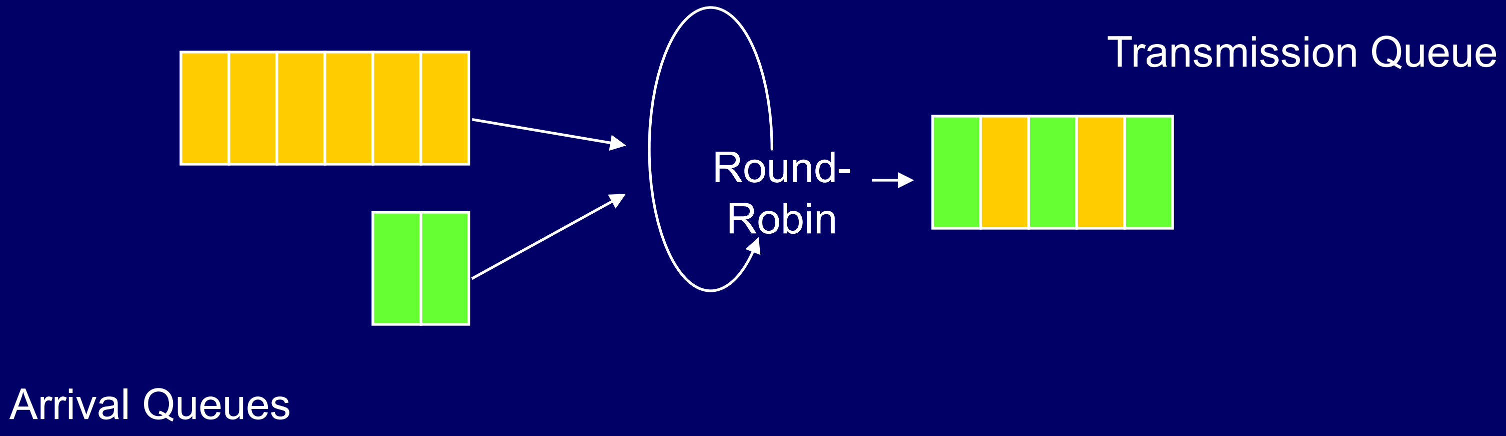
End-to-End Activations

- Resource Management Challenges



Fair Queuing Code for an A.N.E.

- Discriminates between "flows"
- Separate queue for each current flow
- Queues are serviced "round-robin"



Research/Engineering Issues

- What is the relationship between the EE and the NodeOS?
  - What can A.N. applications request?
  - How does NodeOS mux EEs?
- What is the language used for loading disciplines?
  - Per-EE (PLAN code generates Netscript?)
  - RSVP interpreted by A.N.E.?

6. Security

- Models for Security and Safety
- Policy Enforcement
- The Secure Active Network Environment (SANE)
  - AEGIS Secure Bootstrap
  - Local Policy Extension

Security and Safety

- Safety: Good guys can make mistakes...
- Security: Bad guys can program too...
- Network Infrastructure is *shared*
  - it MUST work (telephony as example)
- Can we get **FLEXIBILITY** *and* **SECURITY**?

Challenges: Safety & Security

- Safety: Accidents; Security: Malice
- Specification of goal (@30,000 feet!):
  - *Right* Information to
  - *Right* Place at
  - *Right* Time
- Insecurity: Deviation from goal
  - e.g., information to *wrong* place

Right information/Right place

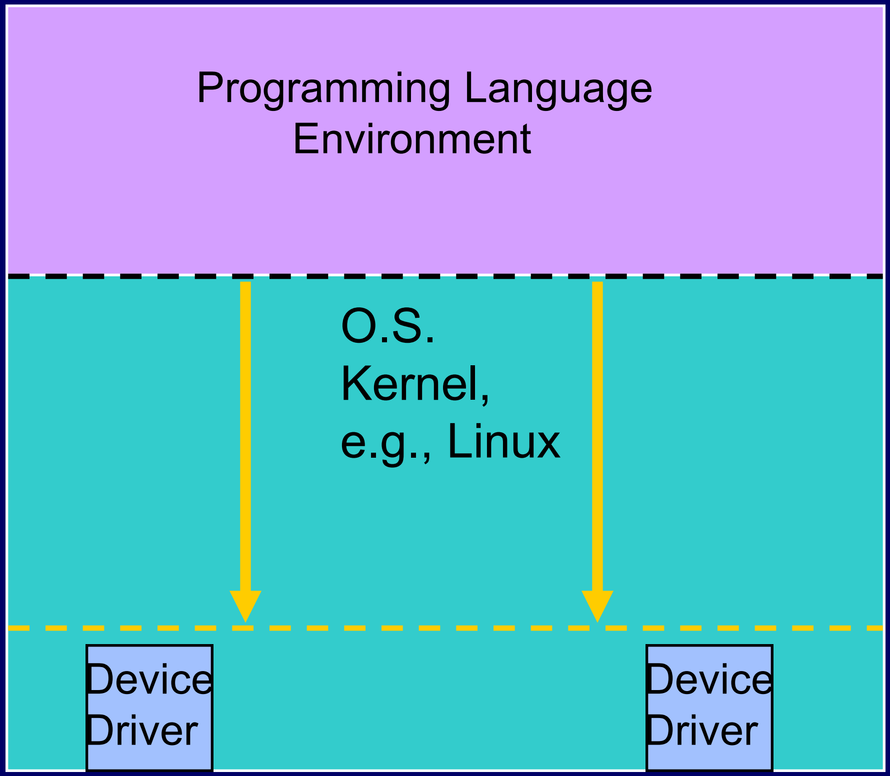
- Requires identifying information units
- Requires identifying places
  - e.g., locations, personnel, etc.
- Requires *security association*
  - e.g., per-place *password* encrypts info.
  - deny information to other places
  - cryptographic protocols: good progress

Right Time (the tricky one)

- Late information may be useless
- Basis of *denial of service* attacks
- Requires identifying *real* times
- Languages have no time semantics
  - gettimeofday() in C/Unix world
  - is ML better? (Dannenberg's *Arctic?)*

How do we control programs?

- Safety & Security: P.L., O.S. or hybrid?

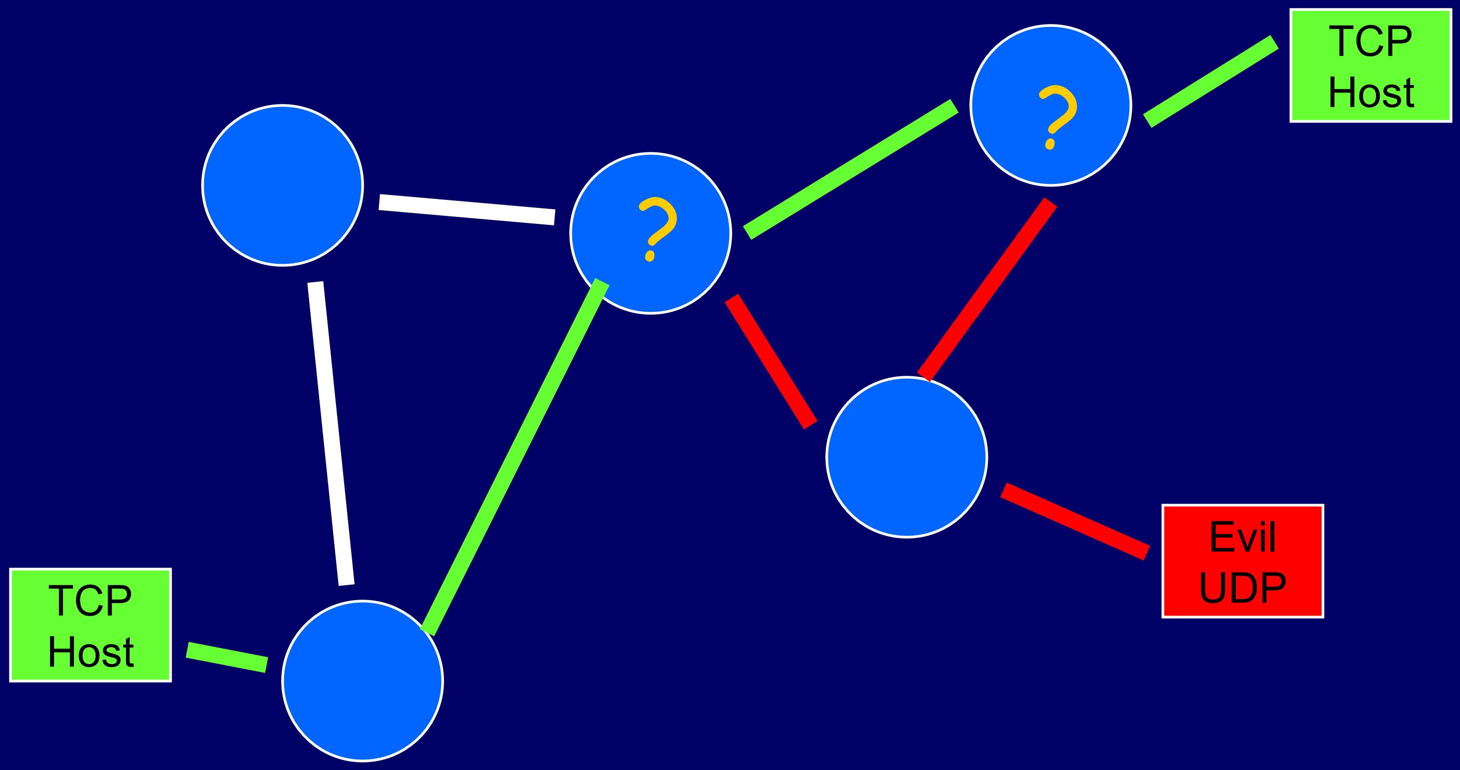


Example: Denial of Service

- Easy to protect server hosts
  - Resource domains, interrupt masking, firewall shielding on host *itself*
- But service is unprotected between client and server site
- This problem *must* be solved with network-embedded functionality

Denial of Service attack

- Cross traffic in an Internet

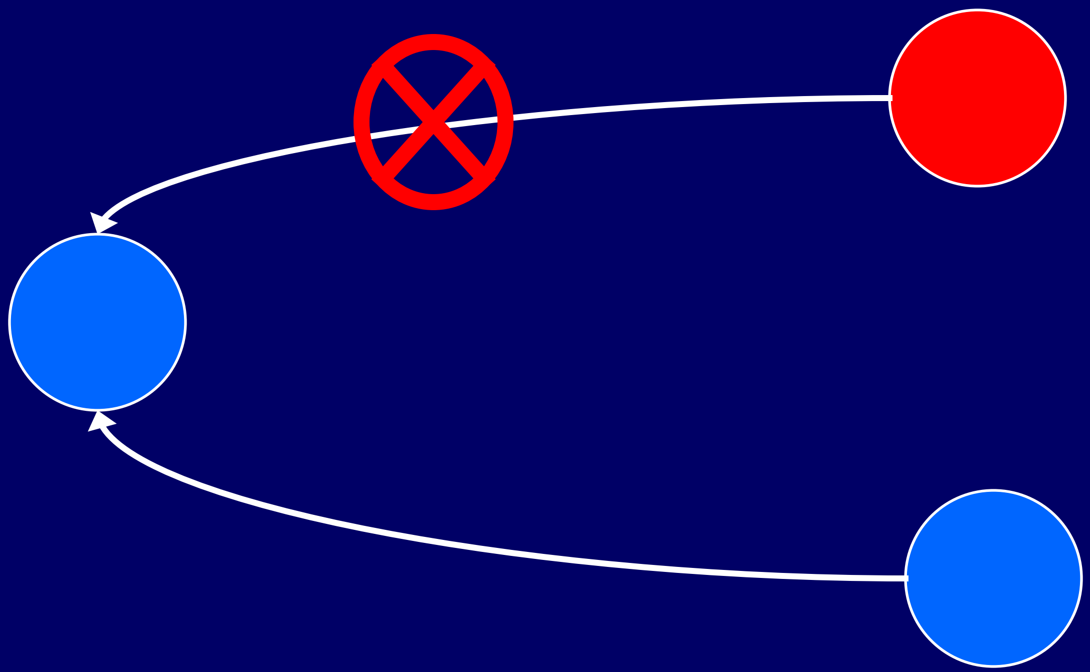


Secure Active Network Environment

- Demonstrates active packet programming
- Extends ALIEN security model into the network with cryptography
- Guarantees no corrupted component
- Allows recovery of failed components
- Enables trust relationships between nodes
- Allows authentication of switchlets

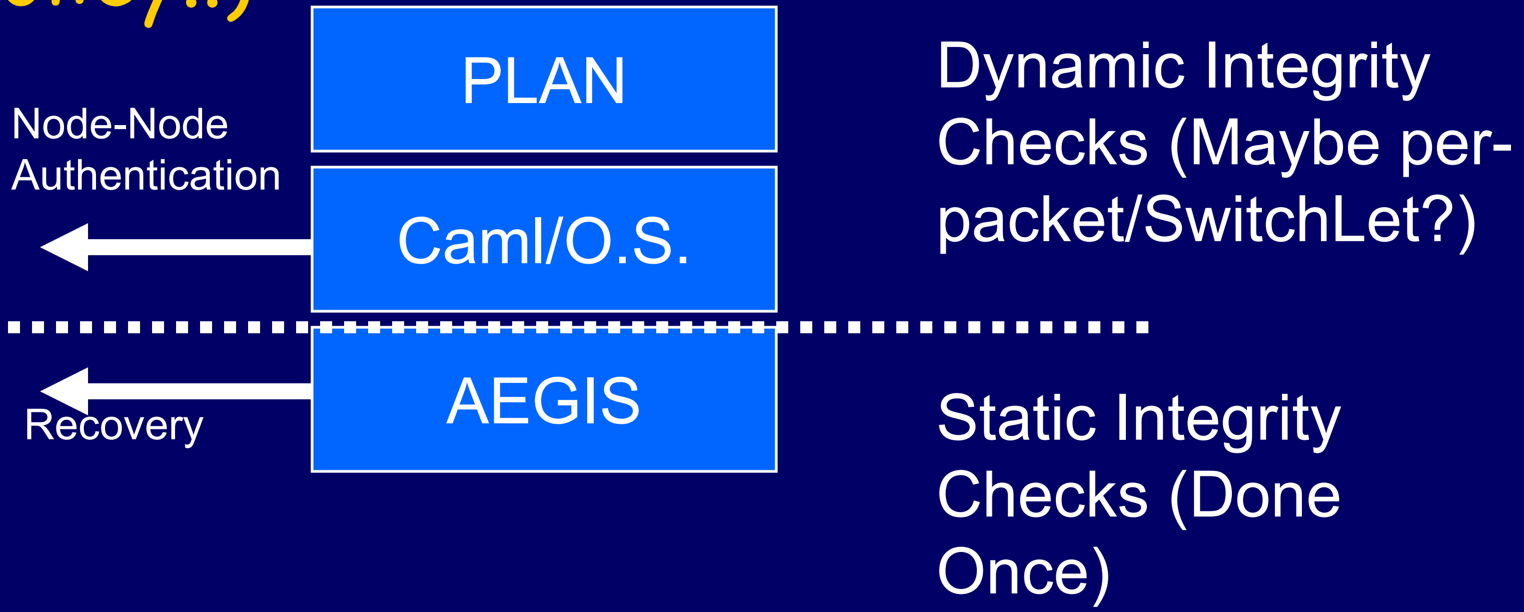
SANE Security Model

- Only process packets from trusted hosts



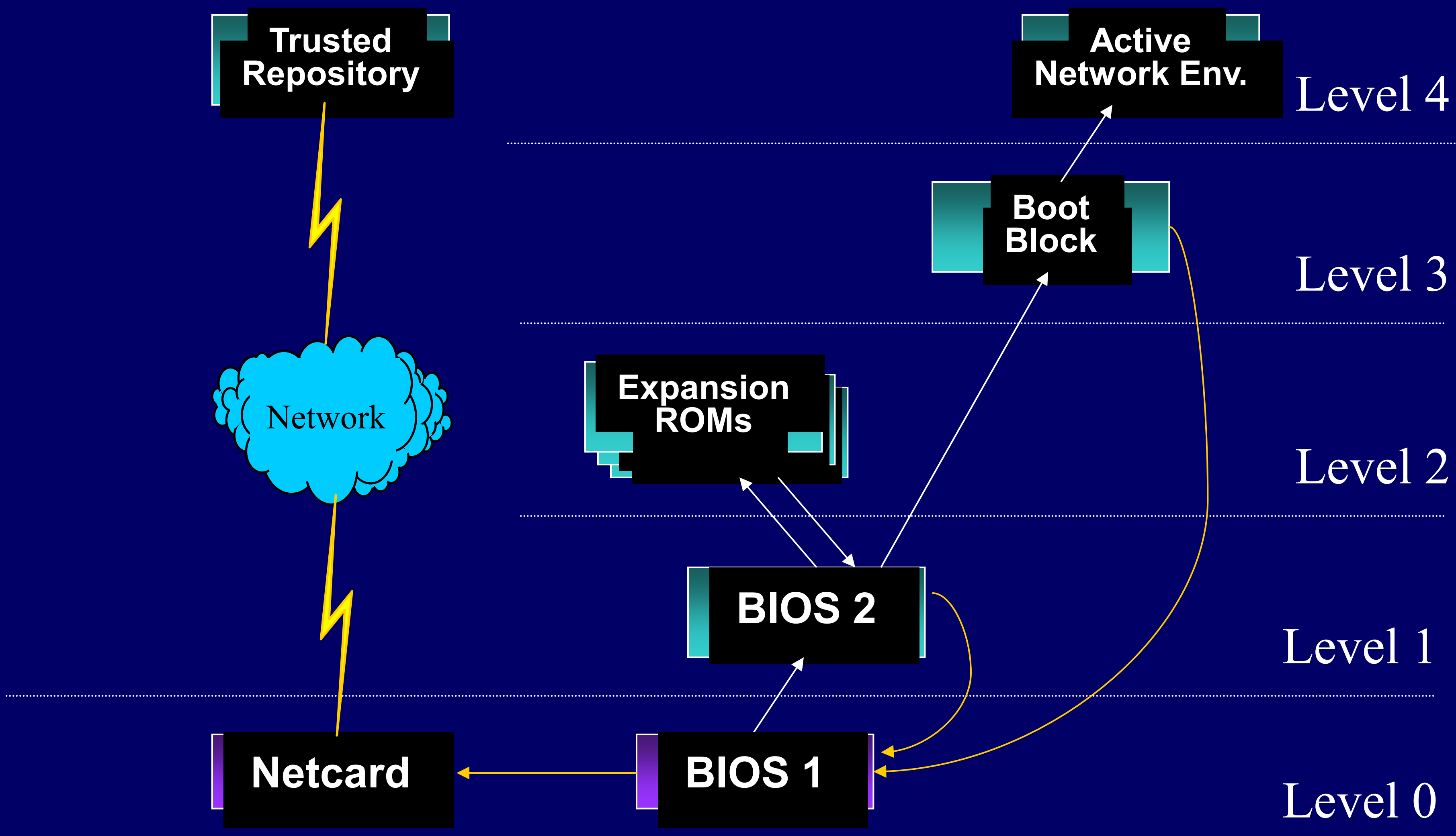
Secure Active Network Element (SANE)

- "Trust, but Verify" (U.S. Nuclear Policy..)

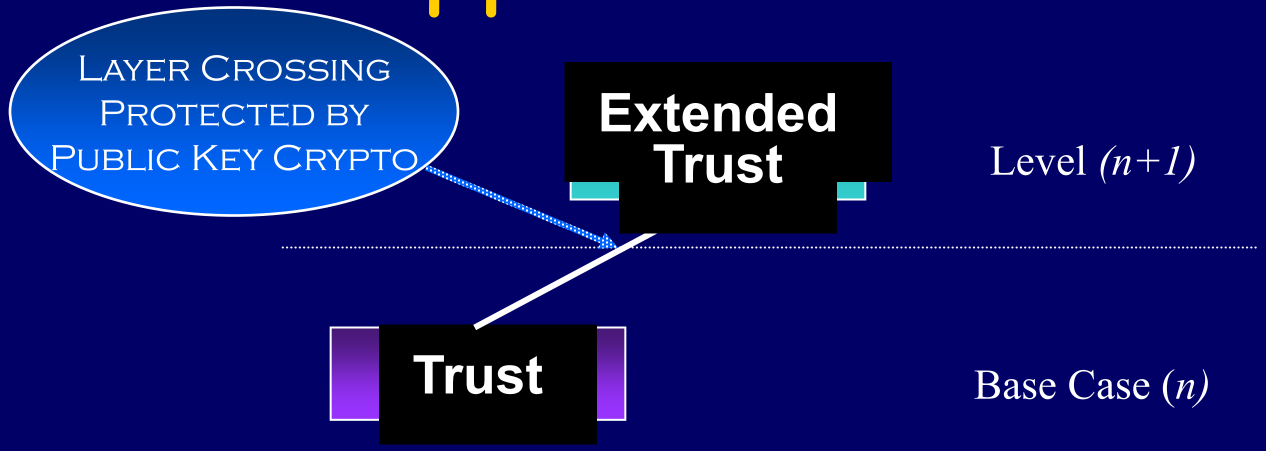


http://www.cis.upenn.edu/~waa
http://www.cis.upenn.edu/~angelos

AEGIS Architecture

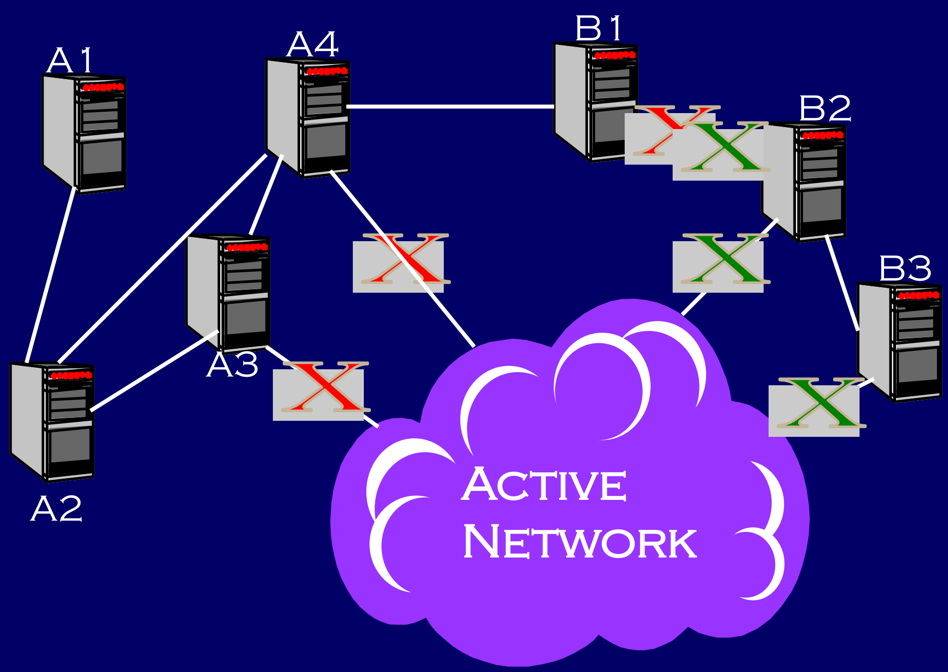


Approach



- Integrity and Trust Must be "*Grounded*" at the Lowest Possible Point.
- Chaining Layered Integrity Checks (CLIC) Extends Trust Beyond the Base Case.

Mutually Suspicious Nodes



- Nodes Authenticate their Neighbors
- Establish Trust Relations with Peers (PolicyMaker?)
- Use Trust Relations to Solve Existing Problems (eg. Routing)
- Optimize Authentication

Node to Node Authentication

- Once at Boot Time, Periodically Thereafter (Crypto “heartbeat”)
- Modified Station-to-Station Protocol (Well Known and Understood)
- Key Can be Used to Authenticate on a Hop-by-Hop Basis, Encrypt Sensitive Information
- Make Traffic Analysis <u>Hard</u>

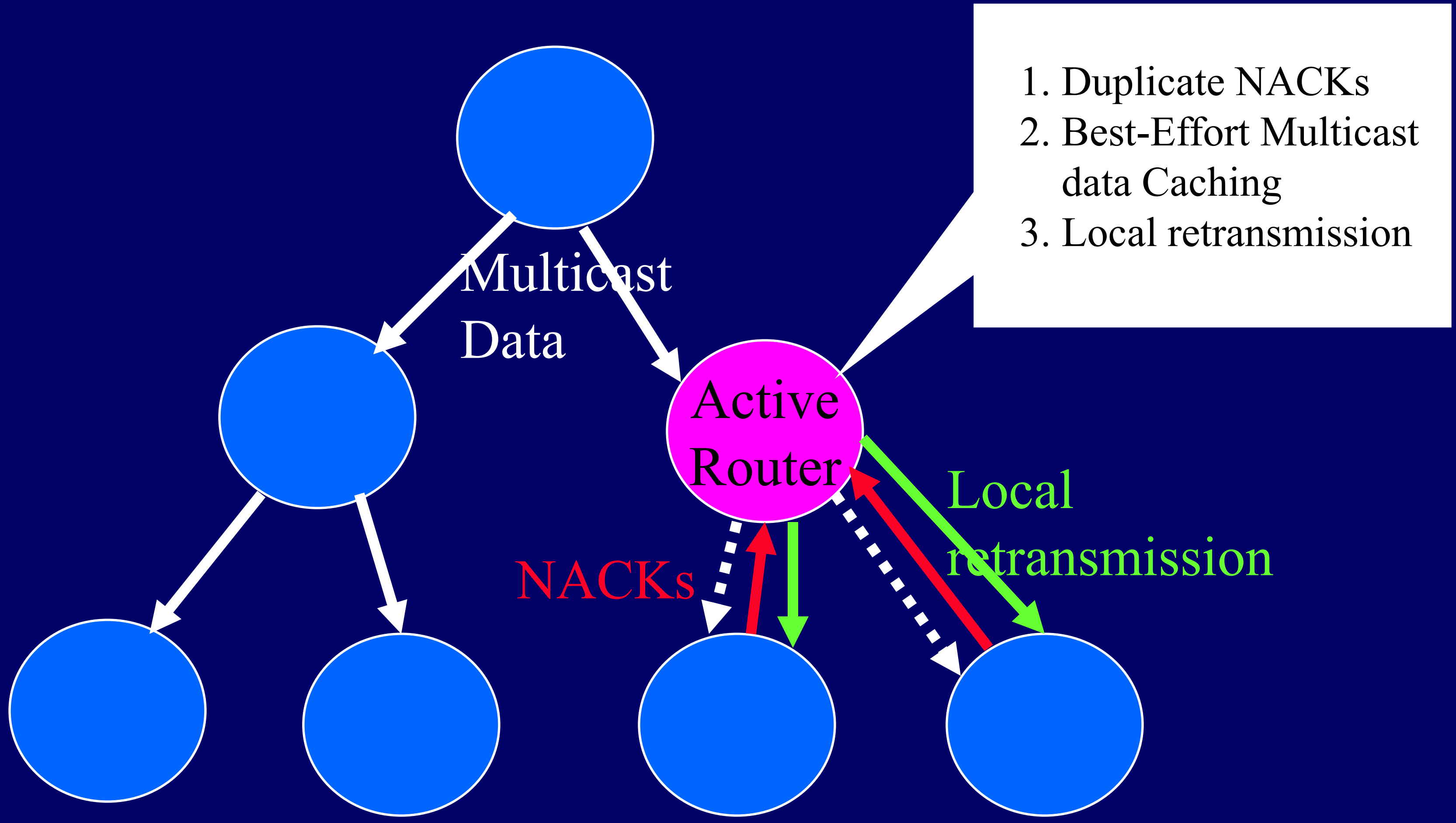
7. Applications

- Active Reliable Multicast
- Active Bridging
- Protocol Boosters
- Active Congestion Control (ACC)
- Active Router Control (ARC)

Active Reliable Multicast (ARM)

- Reliable Multicast plagued by "ACK implosion" when an error occurs
- Retransmission expensive
- In MIT's ARM, Active Elements are embedded in the multicast tree (not all tree nodes need be active for ARM to work)

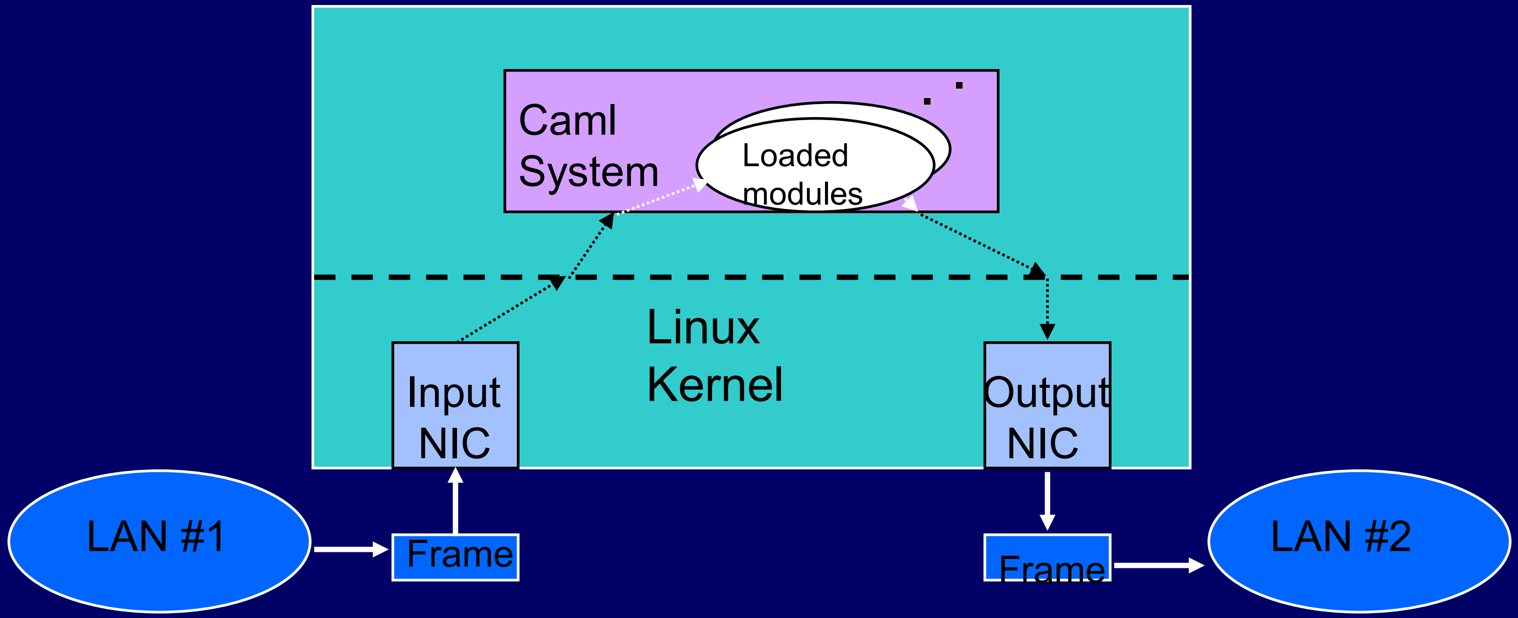
ARM techniques



ARM Advantages

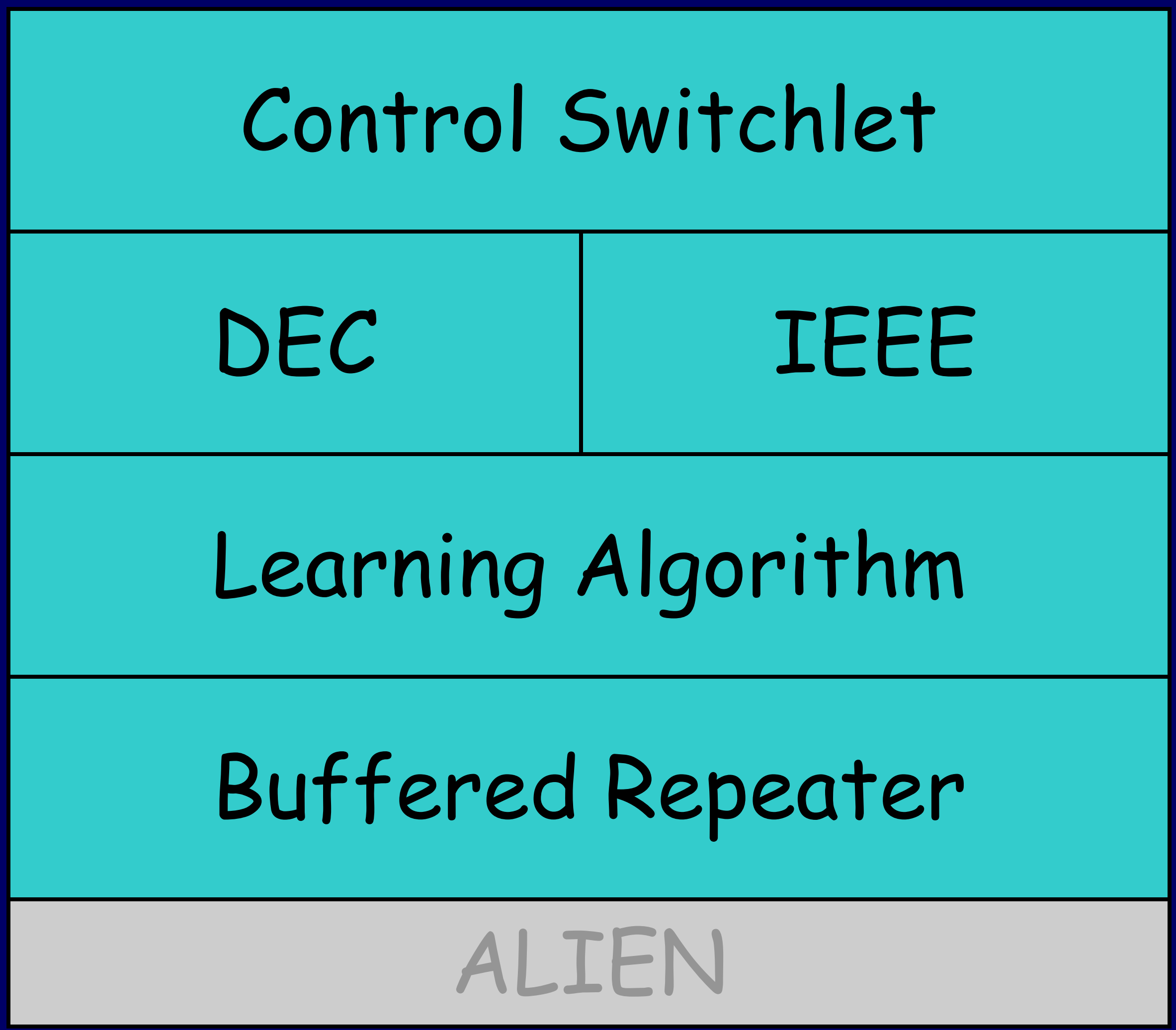
- Simulation shows performance better than Scalable Reliable Multicast (SRM)
- Duplicate NACK suppression (as in Bashkow and Sullivan's ChoPP) reduces load further up the multicast tree
- Cache and local retransmit reduce bandwidth needs

Active Bridging (Scott Alexander, *et al.* Proc. SIGCOMM 1997)



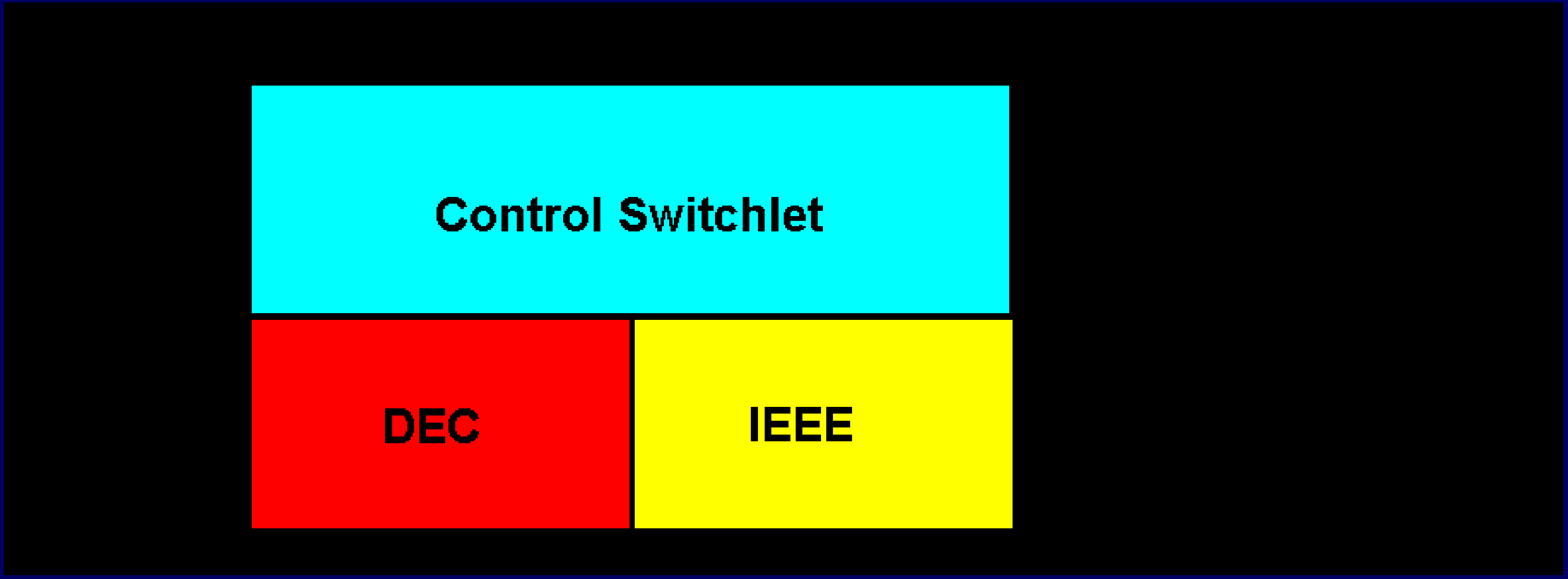
http://oilhead.cis.upenn.edu/~salex

Active Bridge Module Architecture



Automatic Protocol Transition

- Demonstrates dynamic reconfiguration
  - Old protocol
  - New protocol
  - Control Switchlet



Active Bridge Performance

- 58 - 60 Mbps vs. 86 Mbps for C buffered repeater (over 100Mbps Ethernet)
  - Threads and scheduler
  - (Kernel crossings)
- Protocol transition < 0.1 sec
  - vs. 30 sec start up time for IEEE algorithm

Analysis - <u>ttcp</u> throughput

□85.74Mbps □136 μs per packet

□57.73Mbps □202 μs per packet

□□66 μs per packet is cost of Bridge

□54 μs is Bridge processing

□□12 μs "data formatting" (C to/from Caml)

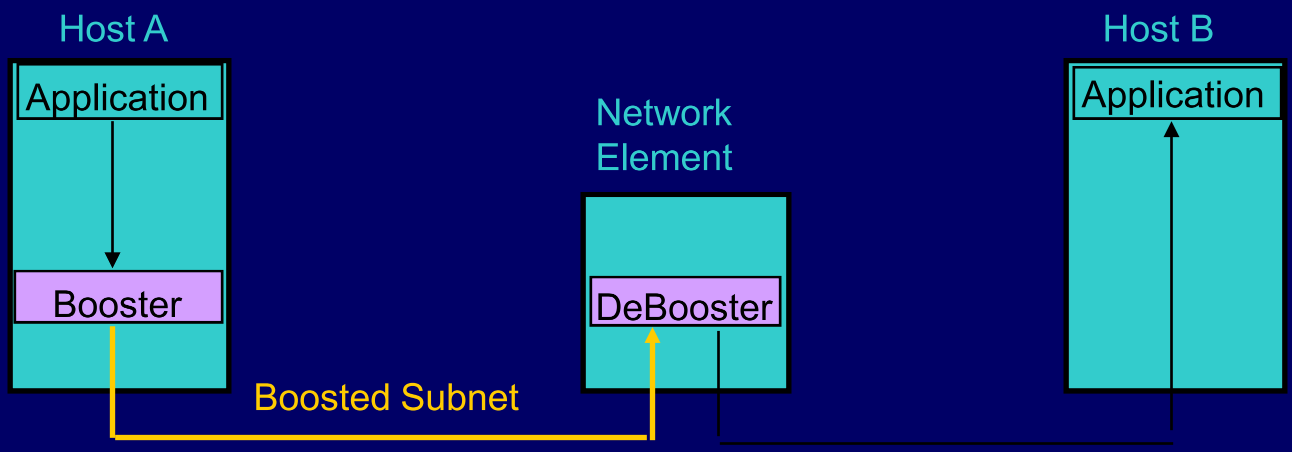
□Language runtime imposed overheads

Lessons from Bridge

- Performance at *ca.* LAN speeds
- Incremental Loads:
  - Buffered Repeater
  - Self-Learning
  - Spanning Tree Algs. (DEC & IEEE)
  - Automatic STA Transition in <0.1sec
  - Recovery from module failure
- http://oilhead.cis.upenn.edu/~salex

Protocol Boosters

- Protocol Elements added ' 'as-needed' '
- Example of "optimistic" design method
- Useful to maintain common case



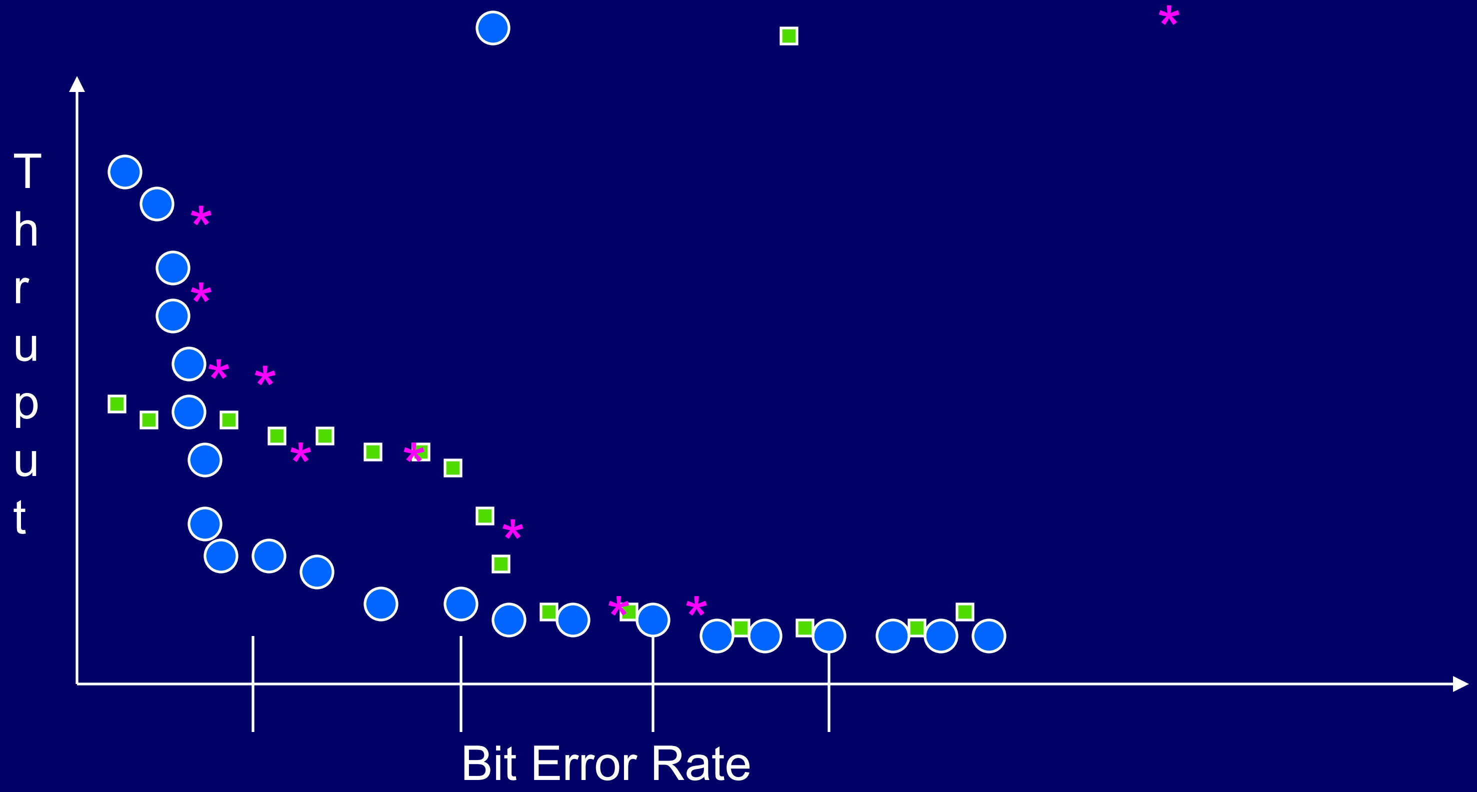
Examples

- Implemented over IP on FreeBSD
  - Encryption Booster
  - Compression Booster
- FEC Booster at Bellcore
- Hardware Support: The P4*

*see http://www.cis.upenn.edu/~boosters/boosters.html

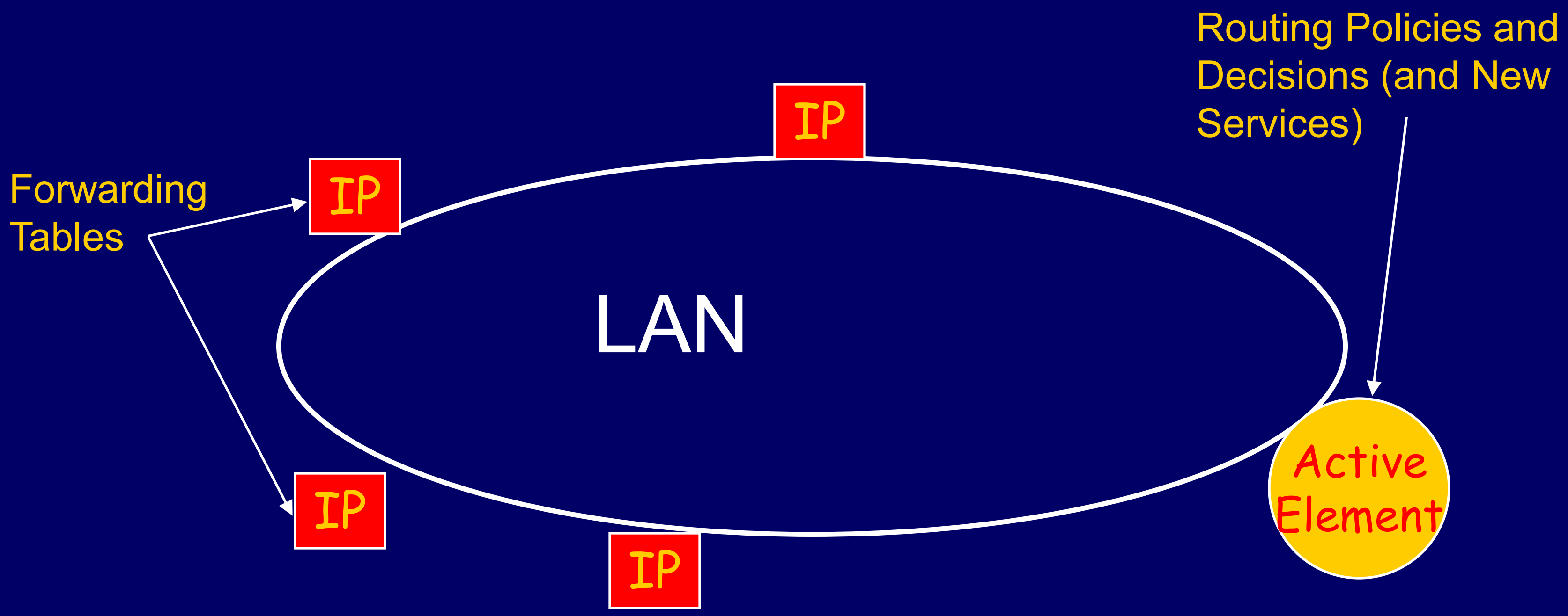
Performance Potential:

- Thruput: TCP, TCP/FEC, Hybrid



Active Router Control (ARC)

- IP Router/Forwarders co-located with Active Elements:



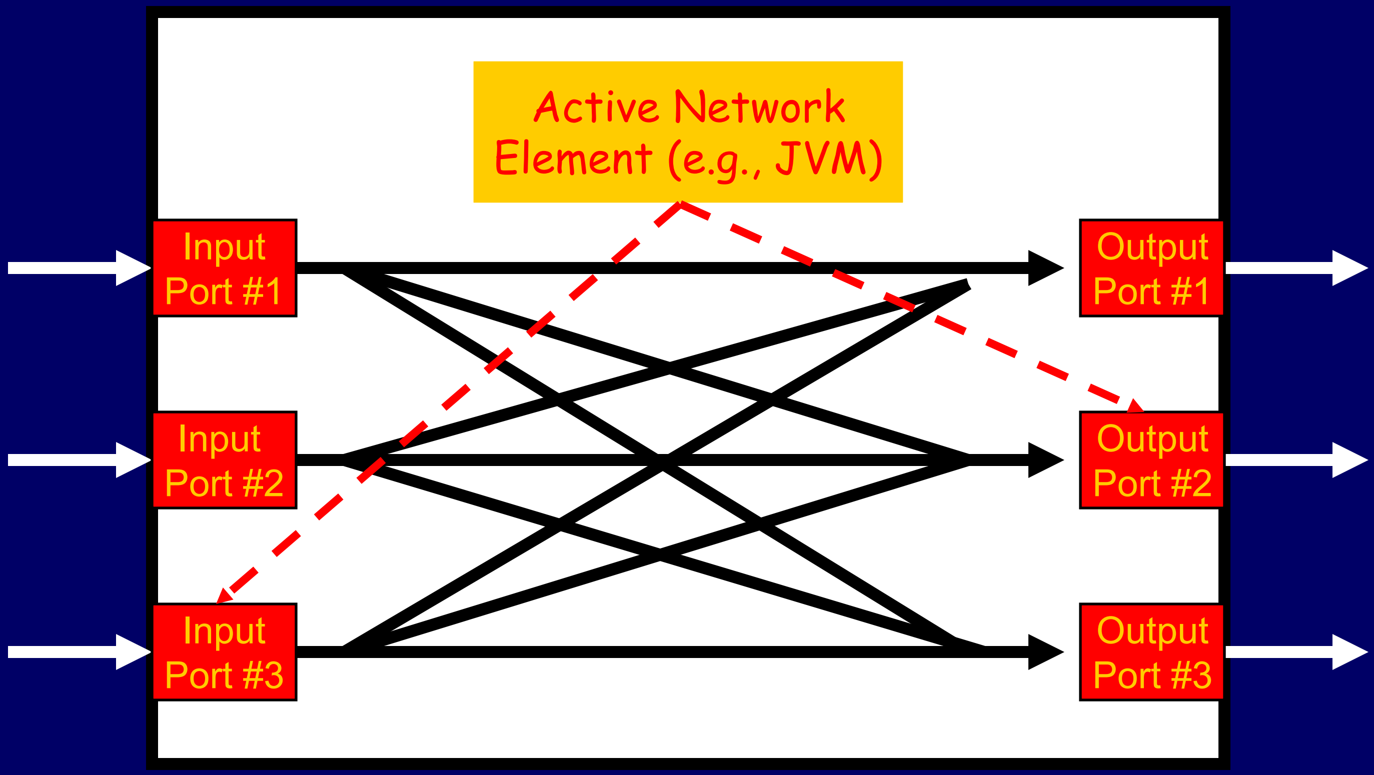
Implementation of ARC, I

- Early experiment by Bill Marcus
  - Bellcore protocol booster kernel on P.C.
  - Control Cisco 7000 through policy based routing (PBR) interface
- Current work by Osman Ertugay at Penn
  - PLAN program on P.C. controlling Cisco 3600 through Policy-Based Rting interface
  - Working with 3Com on CB 3500 platform

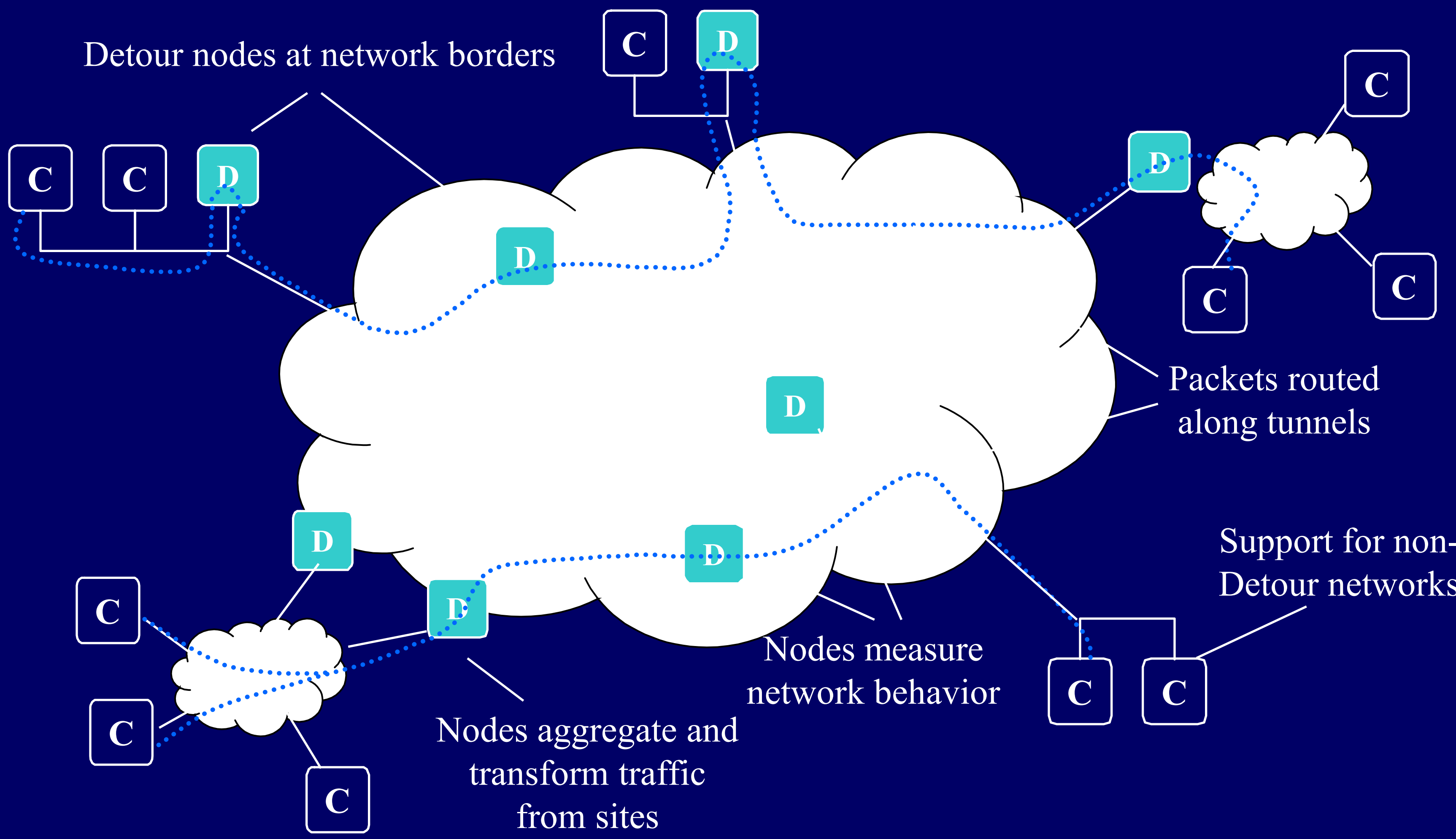
Implementation of ARC, II

- Project by Columbia & Bay/Nortel
  - Netscript on Accelar
- Programmable gateway:
  - Router, firewall, analyzer/shaper, caching server... (boundary smarts!)
  - Investigate SW architecture and HW support

ARC becoming possible in COTS

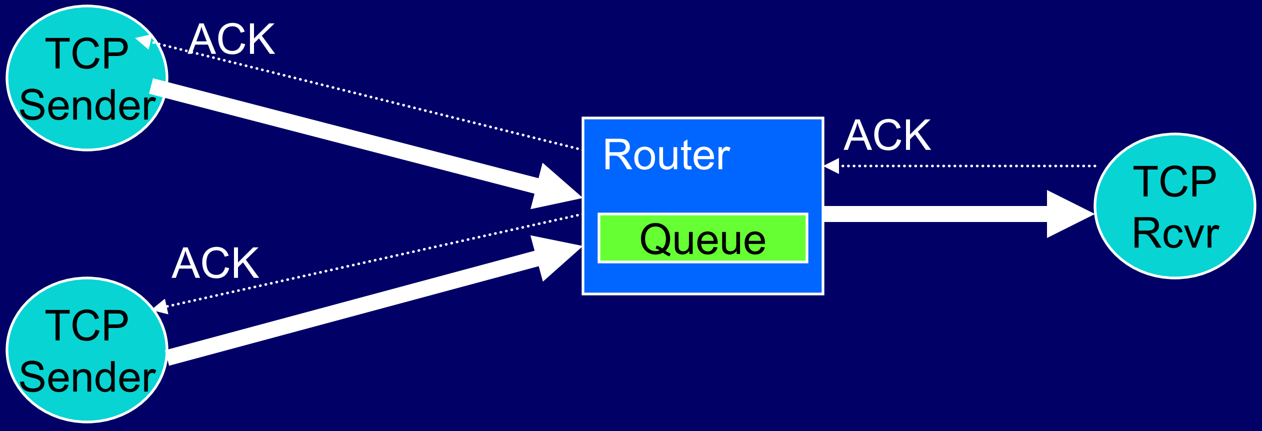


U. Wash Detour Architecture: Cooperating Active Routers



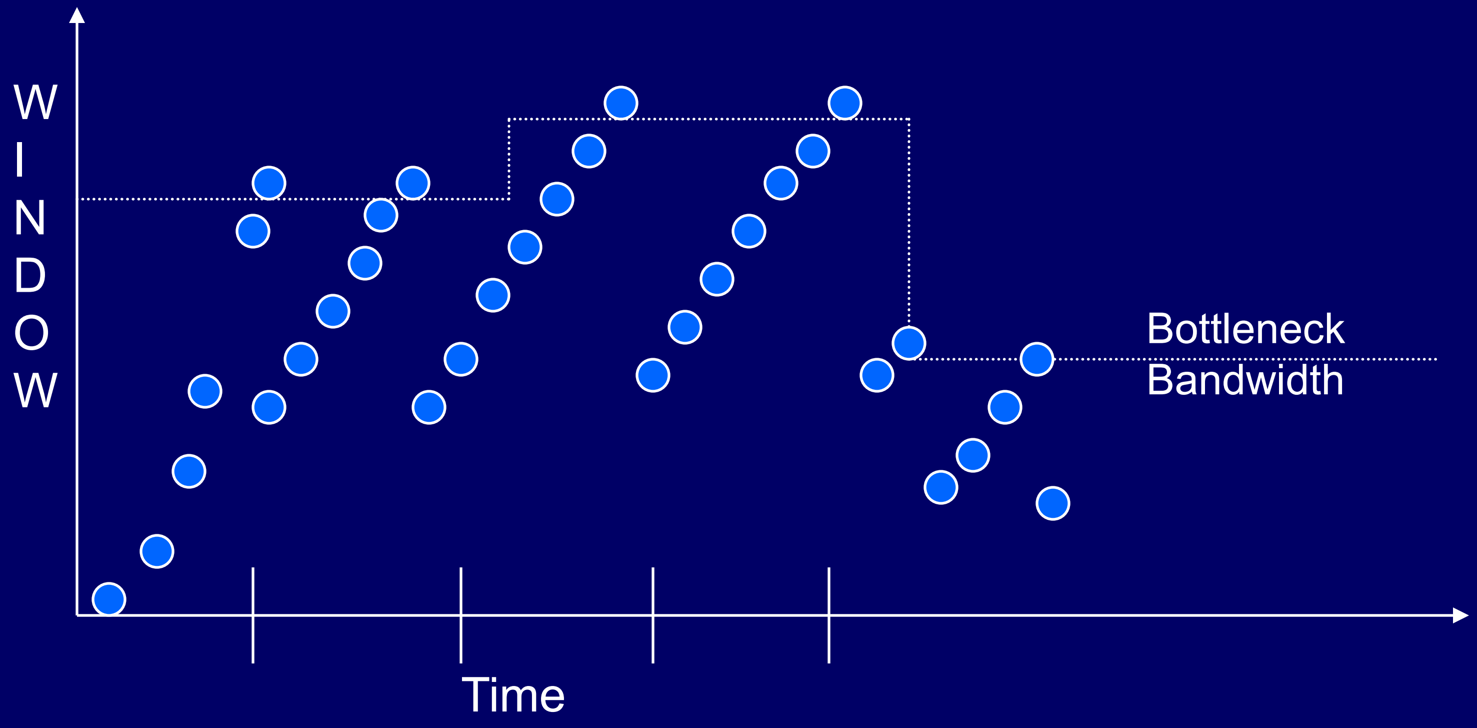
Active Congestion Control (ACC) (Ted Faber, USC/ISI)

- TCP discovers "bottleneck bandwidth"
- Does this with acks/packet loss
- RTT timescale for discovery



Congestion Window Timeline

- Slow-start, then maintenance



ACC models TCP congestion mgmt.

- Drops packets at congested node that would be resent by sender anyway
- Goal of approximating zero delay feedback to sender - defeat latency
- Performance improvements up to 18%
- Good example of network-embedded enhancement for control algorithm

8. Interoperability

- Active Network Encapsulation Protocol (ANEP)
- The ABONE

Interoperability

- Heterogeneous clouds of homogeneity
  - part PLAN, part ANTS, part inactive
  - part Scout, part Nemesis, part SecureXOK
- End to end solution requires:
  - Active border gateways for translation, security domains
  - Communication and resource allocation between execution environments

The Problem(s)

- SwitchWare, ANTS, NetScript, etc.
- Variety of Independent and Important Research Goals
- But, no “ABONE” until they interoperate
- So....let’s make it happen!
- Alexander, Braden, Gunter, Jackson, Keromytis, Minden and Wetherall

Solution: Encapsulation

- Encapsulating Active Network Frames
  - Over Link Layers, IPv6 and IP
- Why header?
  - Find environment for eval.
  - Default processing for missing environ.
  - Non-program information
    - e.g., security headers

What's it look like?

- Format of ANEP Header:

0	8	16	24
Version	Flags	Type ID	
ANEP Header Length		ANEP Packet Length	
Options			
. . . Payload . . .			

Details: Fields

- *Version*: now 1; change w/ANEP header; discard if unknown value
- *Flags*: for V1, only MSB used
  - MSB=0, try to forward w/default
  - MSB=1, discard if TypeID not recognized
- *ANEP Header Length*: in 32 bit words
  - includes options; 2 if no options

Details: More fields...

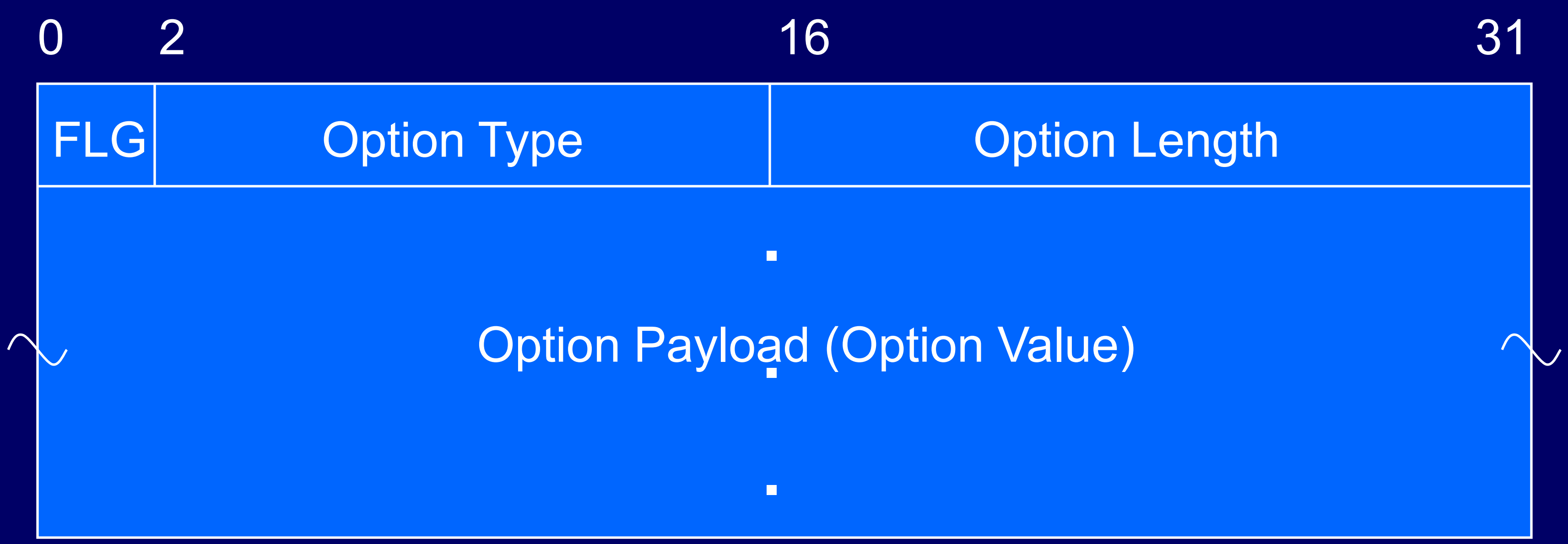
- *TypeID*: evaluation environment for message; 16 bits; values by ANANA
  - ANANA is currently Bob Braden
  - Unrecognized value? Check *Flags* MSB
- *ANEP Packet Length*: Length of entire packet in *octets* (including payloads)
- *Options* length (variable) computed from Packet and Header length difference

Terminology, FYI:

- *Packet*: ANEP Header + Payload
- *Active Node*: Network Element that can evaluate active packets
- *TLV*: Type/Length/Value triple
- *Basic Header*: First two words (8 octets) of the ANEP Header

Options

- Zero or more Type/Length/Value (TLV) constructs
- Follow the basic header. Format:

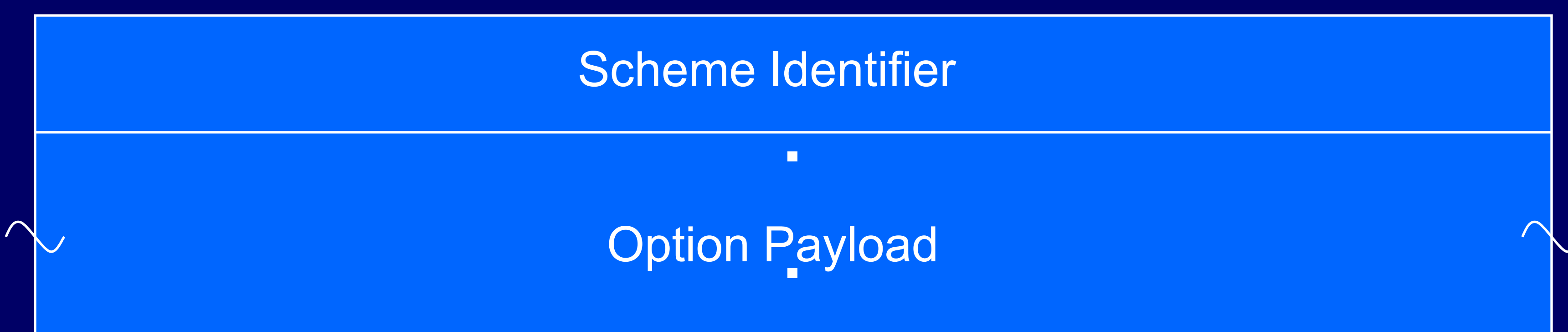


Option Fields

- *Option Type*: 14 bits, used to interpret *Option Payload*.
- Values assigned by ANANA; private when MSB of *FLG* is set.
- Unrecognized value? LSB of *FLG* 0, continue; 1 discard packet. Should log.
- *Option Length*: 16 bits; TLV length in 32 bit words; >= 1.

Option Type Values

- Reserved:
  - 1 - Source ID
  - 2 - Destination ID
  - 3 - Integrity Checksum
  - 4 - Non-Negotiated Authentication
- Format for Source, Destination, N-N:



Source Identifier

- Uniquely identifies sender
- *Scheme Identifier* is 32 bits; identifies addressing scheme to interpret the variable size *Option Payload*
- Reserved:
  - 1 - IPv4 Address (32 bits)
  - 2 - IPv6 Address (128 bits)
  - 3 - 802.3 Address (48 bits) (last two octets 0)

Destination Identifier

- Uniquely identifies destination in the active network
- Same payload option format as Source Identifier

Integrity Checksum

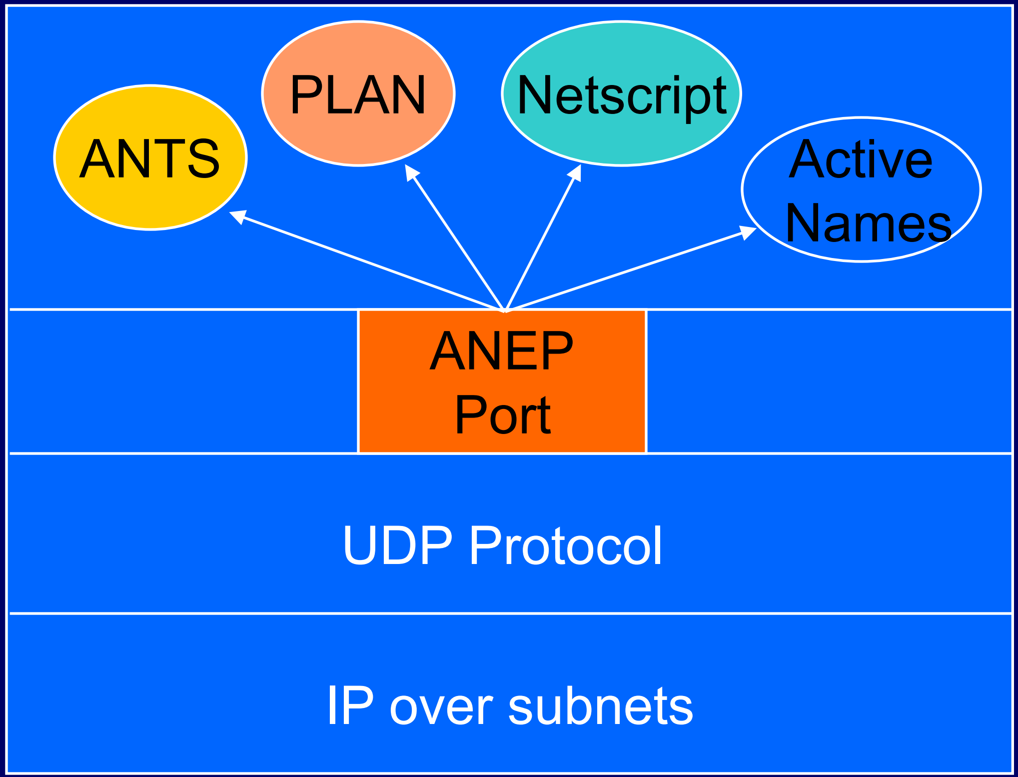
- Detect some packet integrity losses
- 16 bit 1's-complement of 1's-complement sum of the ANEP packet from the ANEP Version field
- Payload zero for computing checksum
- Option length field is 2.

Non-Negotiated Authentication

- Provides 1-way authentication
- No prior negotiation assumed
- Option payload: 32 bit authentication scheme, followed by scheme's data.
- Option length field >2.
- Reserved:
    - 1 SPKI self-signed certificate
    - 2 X.509 self-signed certificate

ANEP demultiplexes to EEs

- Well-known UDP/IP Port for ANEP

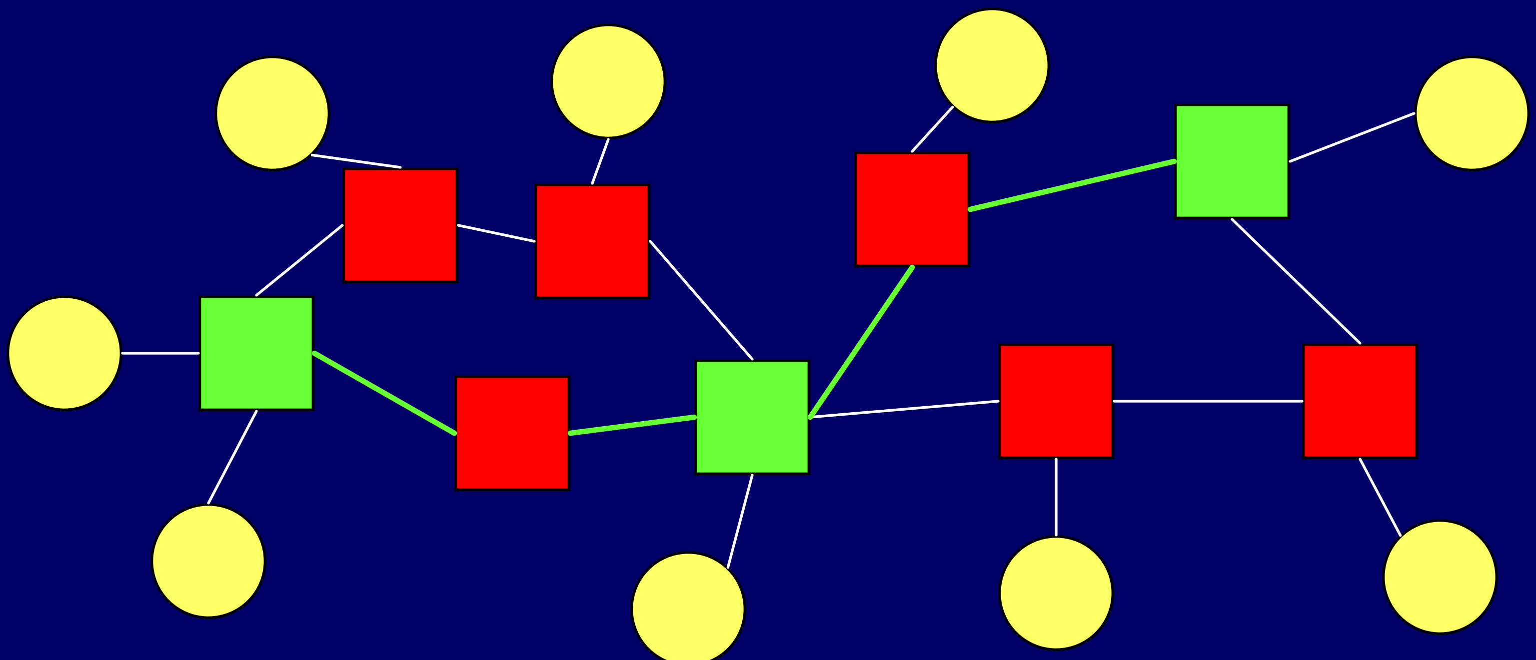


ANEP Summary

- ANEP is not the end, a way to get going
- SwitchWare, ANTS, Netscript operate ANEP
- Interoperability using existing infrastructure

ABONE tunnels over Internet

- Hosts
- IP Routers
- Active Network Elements



Research/Engineering Issues

- Hierarchy necessary to scale
- Extend with ARC<->ARC protocol
  - ARCs will be organized in Admin. Domains
  - Arbitrary ARCs cannot control routers
  - ARCs resemble active firewalls
- At border gateway, need translation/communication between EE's

Summary: Interoperability

- Towards *PLANTScript*
  - Internet -- hook networks together
  - Interactive network -- hook active networks together
- Federated administrative domains
  - No single node OS, API, prog lang Required if system is to *scale*
  - Security, perf. isolation, local decision making, upgrade path, ease of devel.

9. The Future

- Fiber optics and Active Nets
- Hardware Support for Active Nets
- Node Security vs. Network Security
- Deployment and commercialization
  - Computation Over Bandwidth (COB)

Do All-optical nets invalidate Active Nets vision?

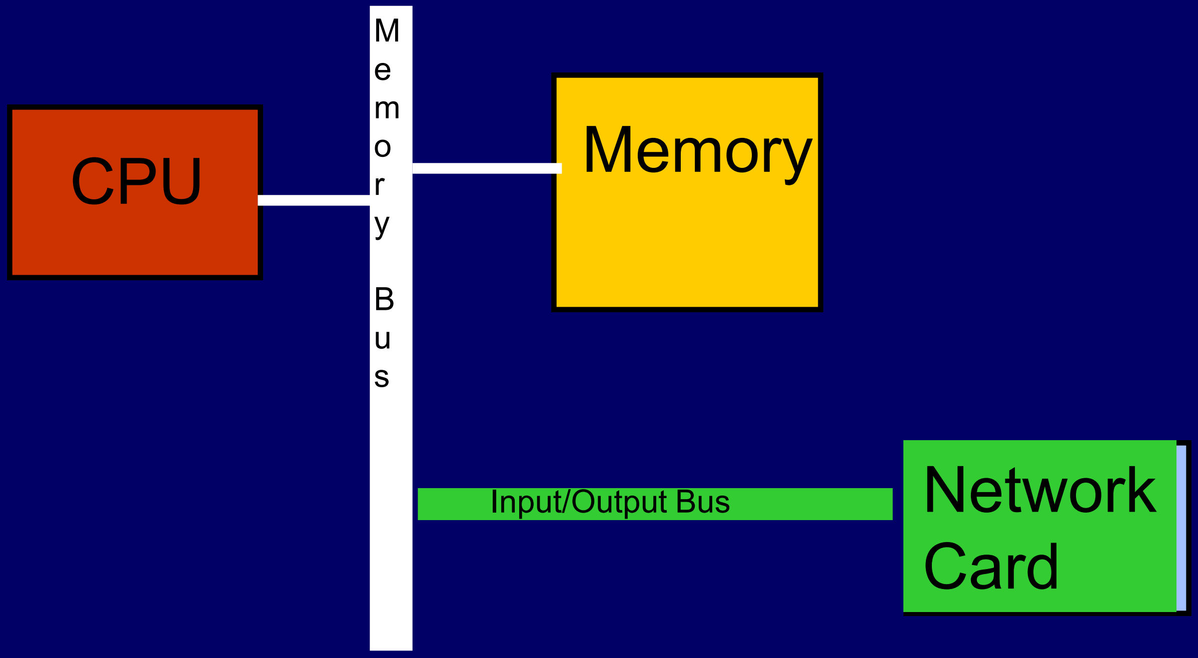
- Well, at a high-level, *no*!
  - ANTS, PLAN, ALIEN fast enough for home/access point/LAN, up to peering point
- But what about *really* exotic speeds?:
  - exponentially-improving CPU speeds
  - exotic technologies, e.g., mediaprocessors
  - or general-purpose CPUs in new archs.?

Some rough arithmetic...

- OC192c SONET is 9.6 Gb/s
- For 64 bit CPU, 150 MW/s
- Clock rates of 500-750 MHz mean:
  - RR moves: 2-3 W/instruction
  - Register file writes likely bottleneck
  - So about 5 instructions/word
  - Can't afford any delays

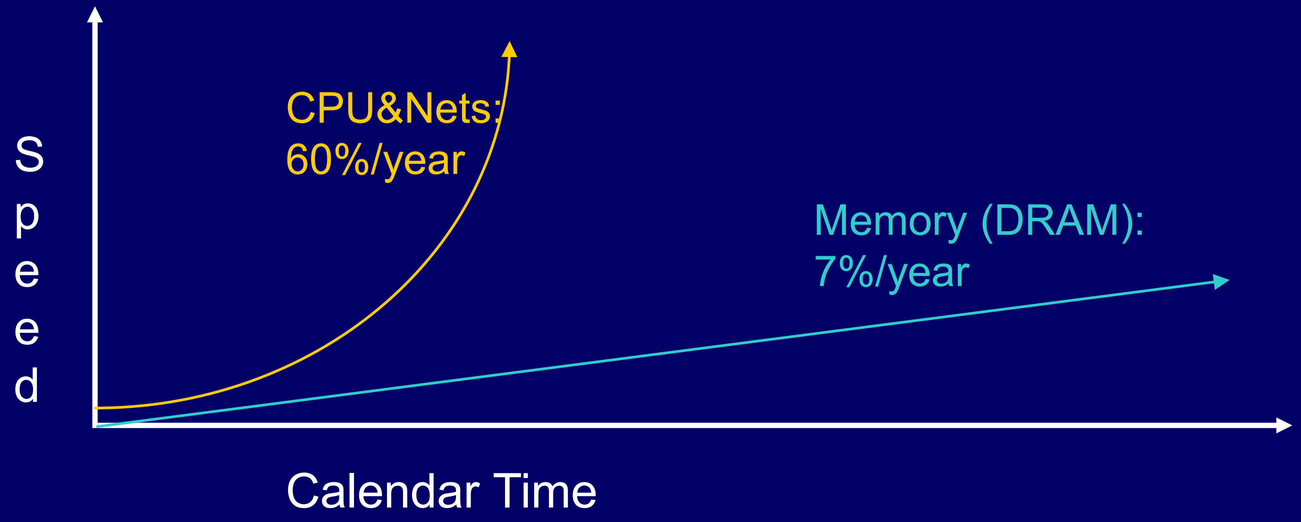
Typical Computing, Memory & Network Attachment

- Architecture:



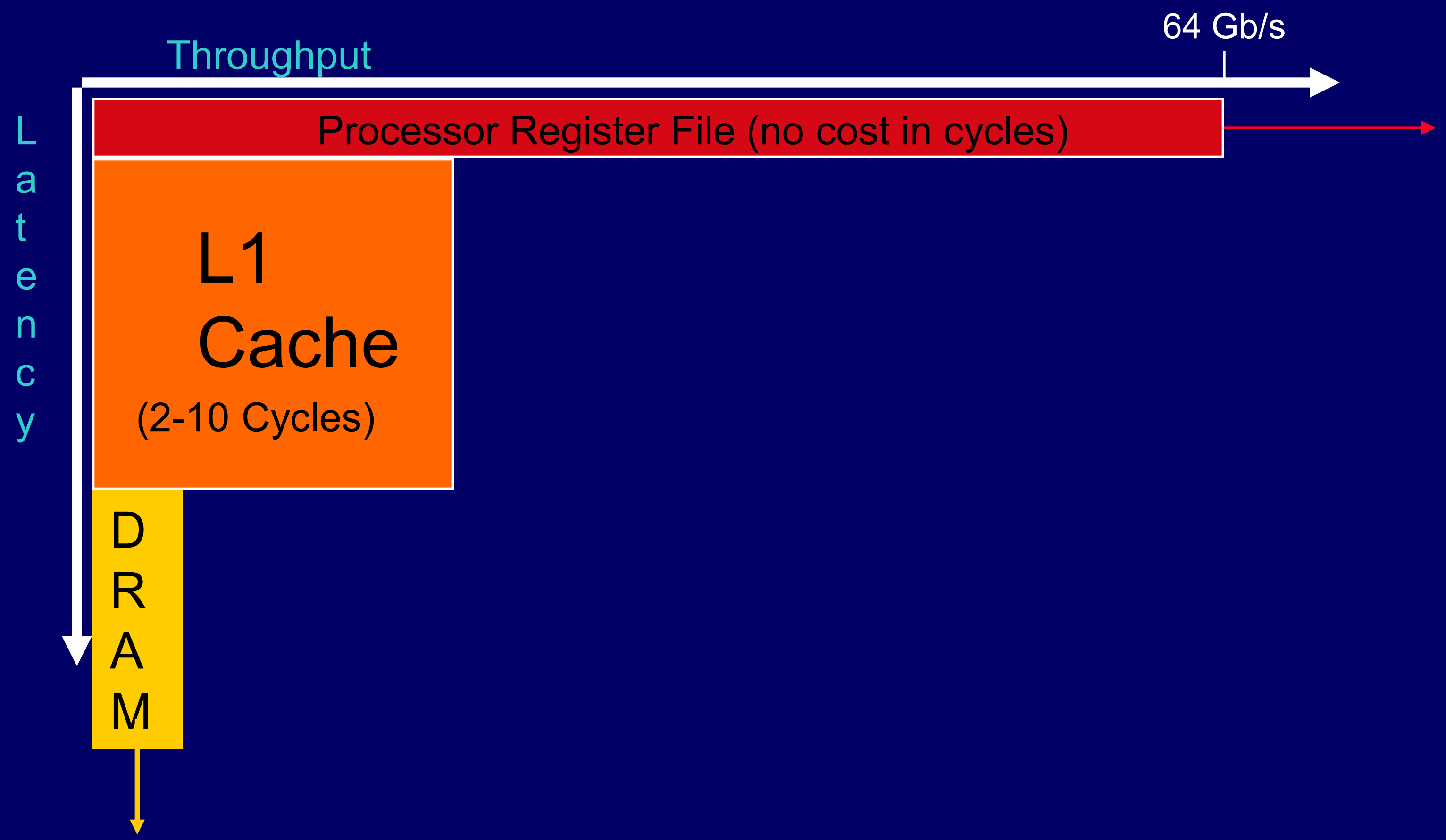
Why this won’t work: mismatched exponentials

- Memory exponential has been <u>capacity</u>

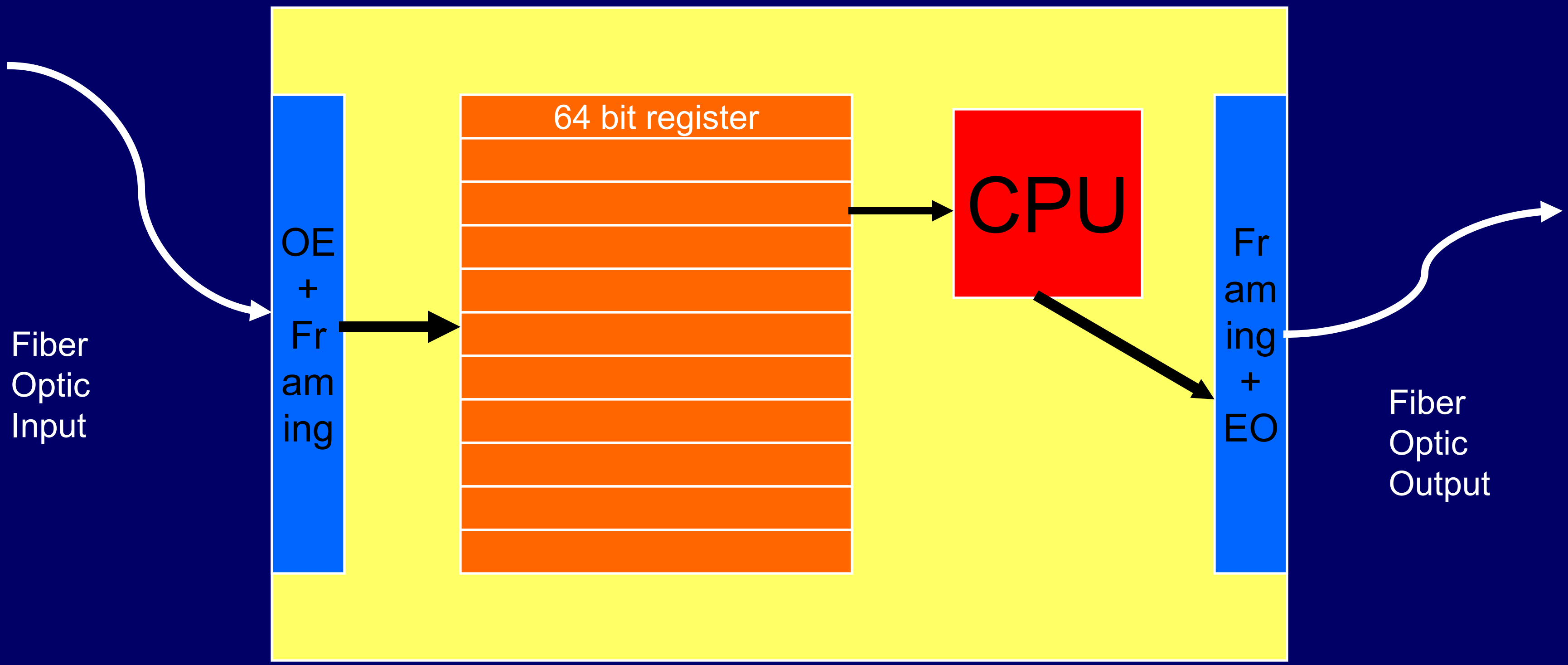


Not throughput!

- Unattractive tradeoffs for networks:

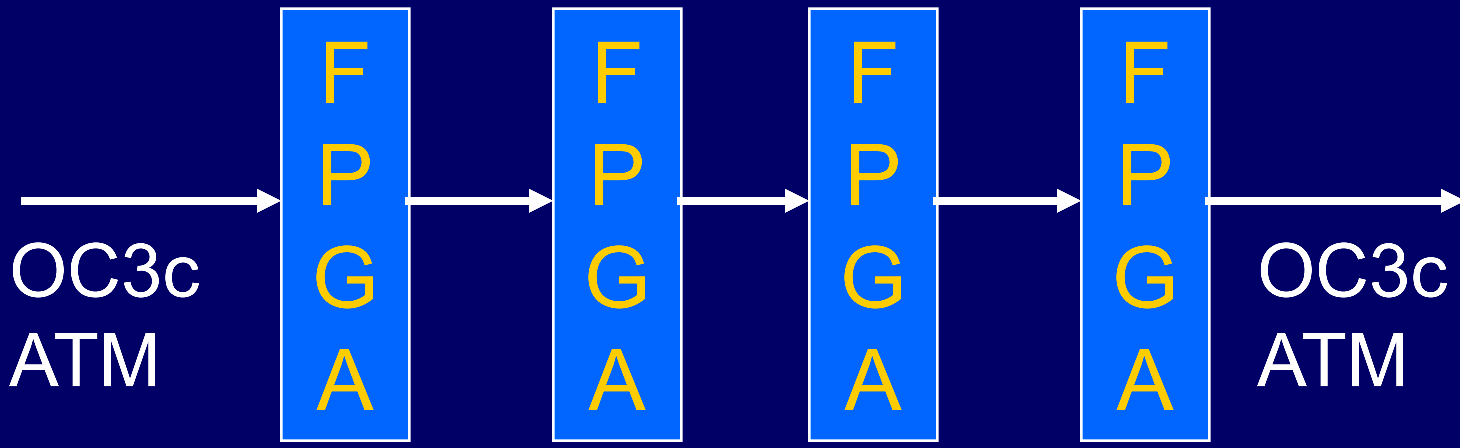


Fiber-coupled processing?



Register-Only Media Processor (ROMP)

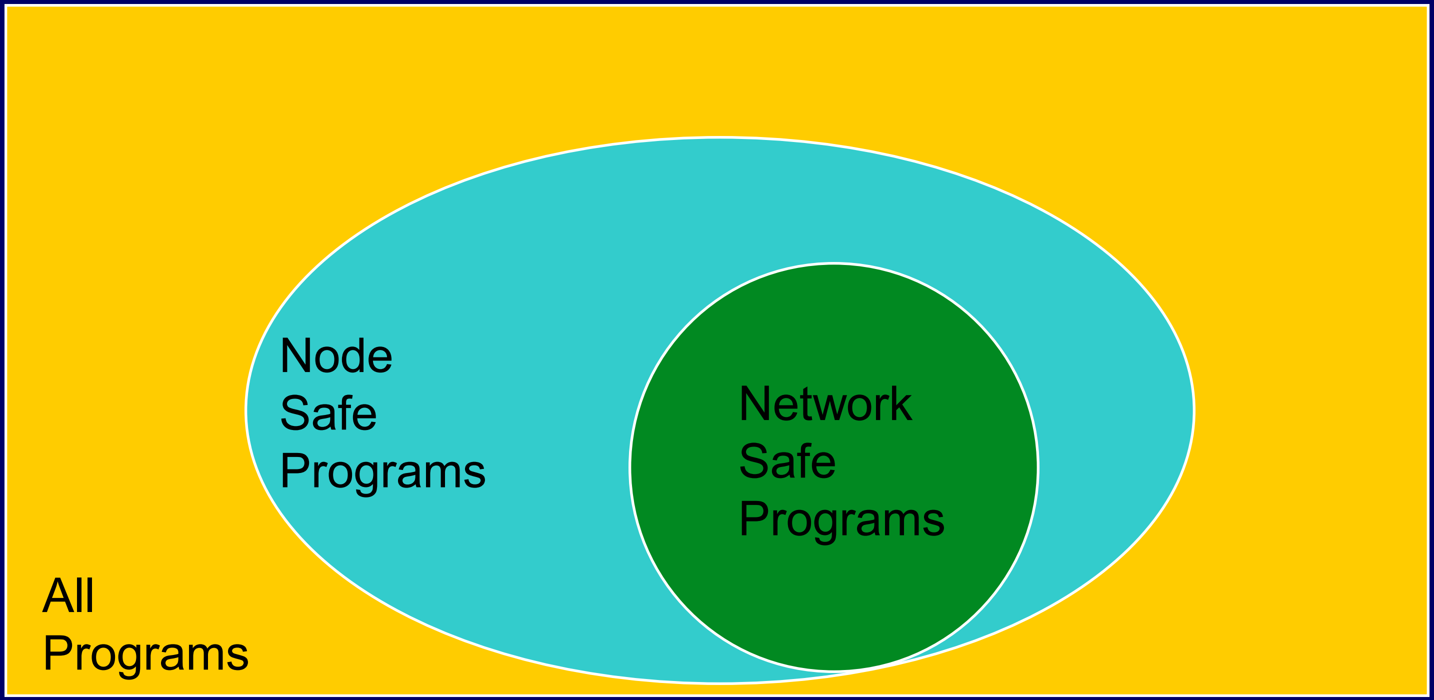
Protocol Processing Pipeline (P4)



http://www.cis.upenn.edu/~boosters

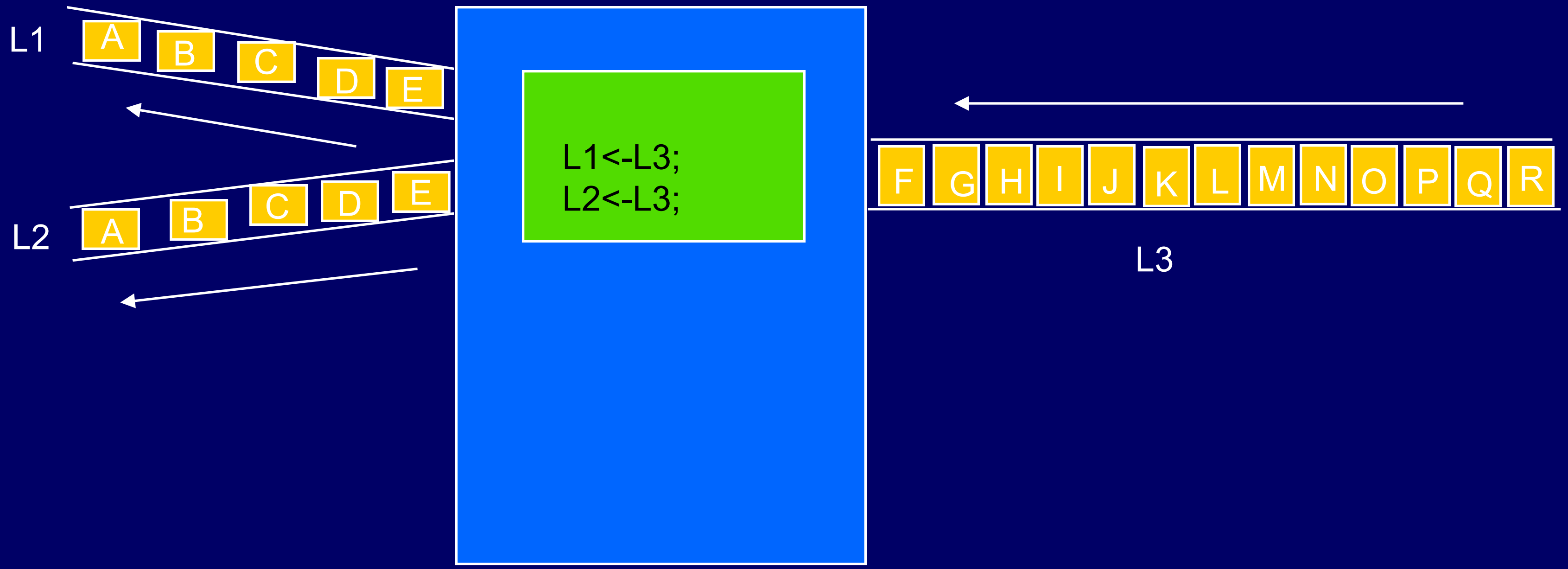
Restricting Programs

- Node safe versus network safe



Example: Local versus Global

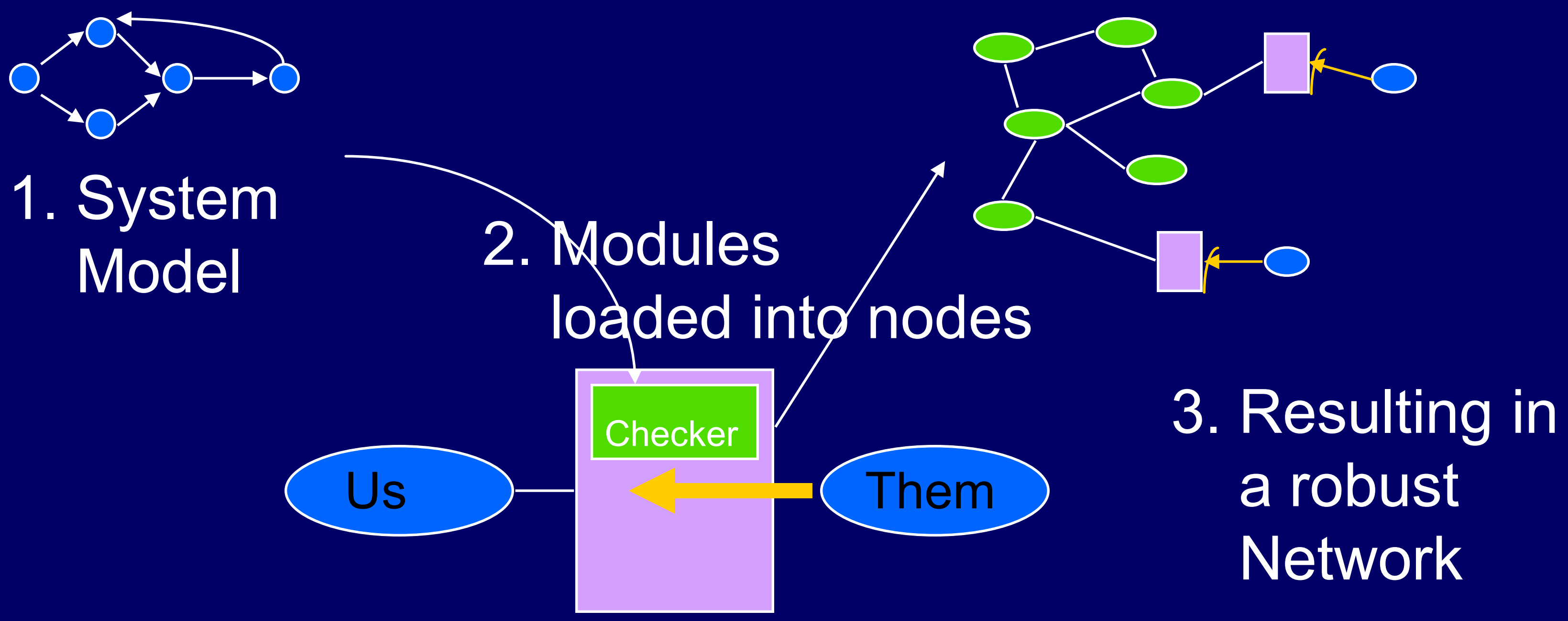
- Program copies L3 (in) to L1, L2 (out)



- Is this “Multicast” Program “safe”?

Model->Modules->Actions

- Syntax, Semantics, Node vs. Network
- Example: Securing a Network



Activation potential at various commercially deployed rates: