Design of a High Performance Active Router

Jonathan Turner, Guru Parulkar (on leave)
Dan Decasper, John Dehart, Sumi Choi, Tilman Wolf,
Bernhard Plattner (ETHZ), Ralph Keller (ETHZ)

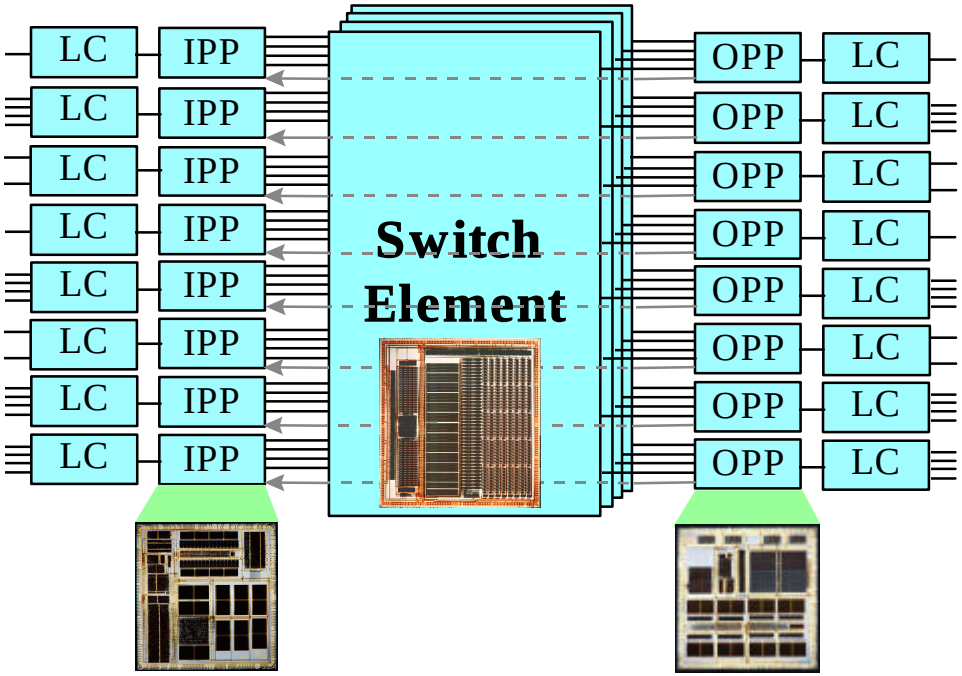
Project Objectives

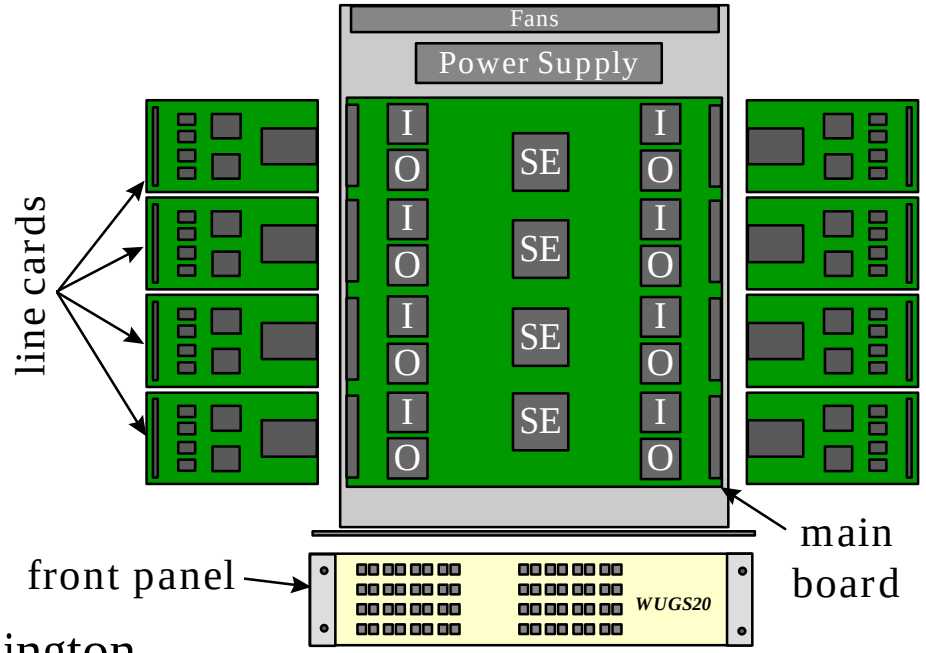
- Demonstrate potential of *high performance* active networking.
- Design and implement a prototype active networking platform.
  - » high performance - active processing at gigabit rates
  - » scalable - prototype capacity of 20 Gb/s - scalable design to over 1 Tb/s
- Develop general methods for high performance.
  - » scalable and extensible hardware platform
  - » software architecture supporting parallel processing
  - » kernel plug-ins for efficient software processing
  - » use of compiled code from distributed code servers
  - » authentication to ensure code integrity
  - » reconfigurable hardware components
- Demonstrate active applications
  - » protocol deployment
  - » topology-aware reliable multicast
  - » video distribution over reliable multicast

Key Technology Components

- Scalable Switch Core
  - » initial prototype 20 Gb/s, 160 Gb/s configuration under construction
  - » built around custom ATM chip set developed at Washington University
- Embedded Processor at each Port
  - » Pentium Processor with 32 MB memory
  - » NetBSD operating system with active network extensions
  - » high performance two port network interface chip
- Kernel Plugins for Efficient Packet Processing
  - » configurable software components in OS kernel
  - » allows application-specific processing of compiled code in kernel
- Distributed Code Caches for On-Demand Plugin Retrieval
  - » non-resident plugins retrieved from hierarchy of trusted code servers
  - » authentication mechanisms used to verify integrity of retrieved code

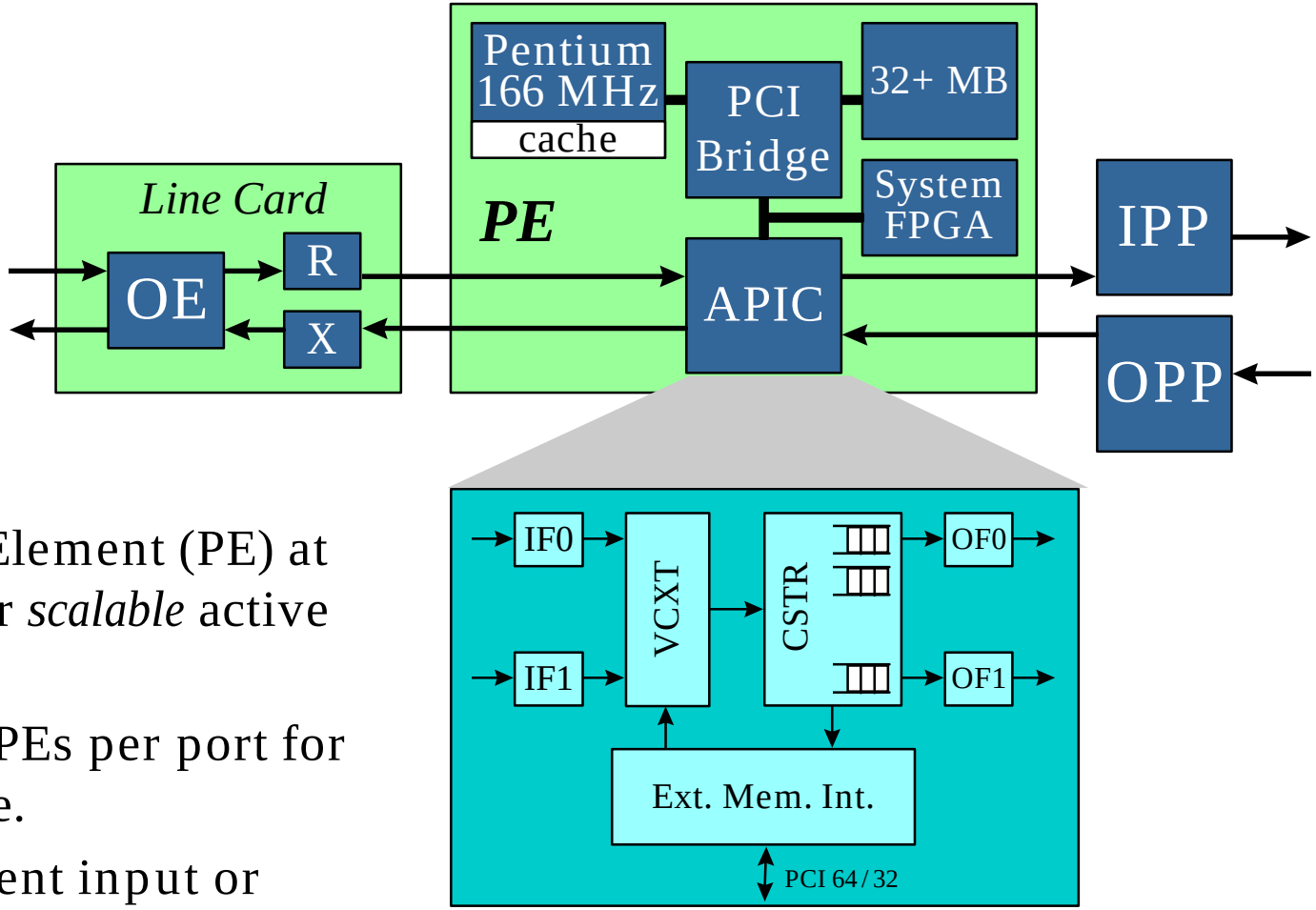
Washington University Prototype Switch





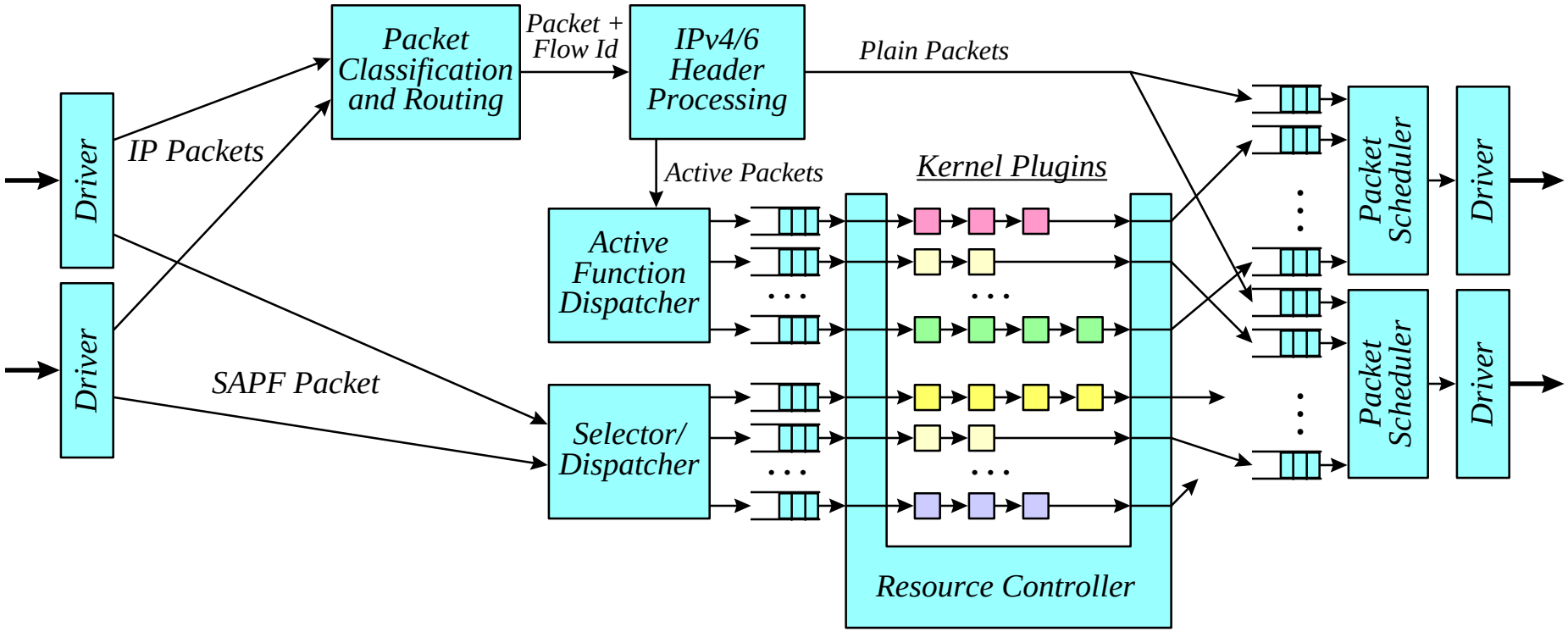
- 20 Gb/s core capacity on single PC board
- efficient multicast support in hardware
- variety of line cards (up to 2.4 Gb/s)
- Copies being distributed as *open research platform* to 30 universities under NSF sponsorship

Active Node Processing Element



- Active Processing Element (PE) at each switch port for *scalable* active packet processing.
- Can *stack* multiple PEs per port for higher performance.
- APIC enables efficient input or output side processing.

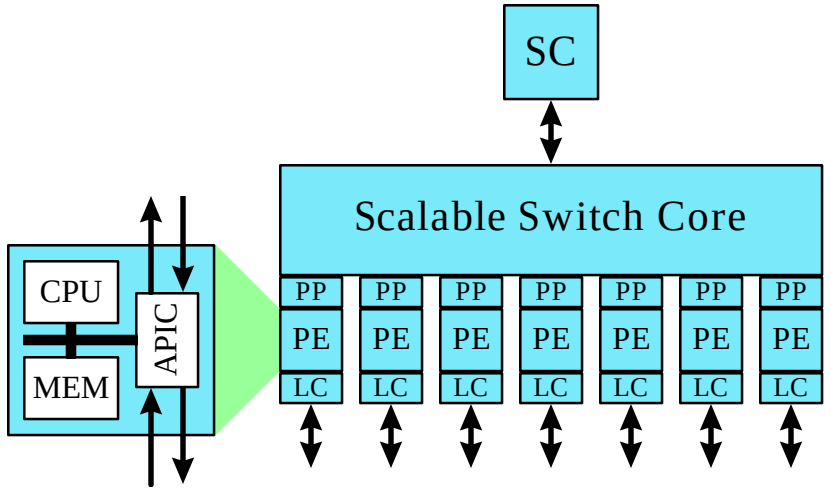
Principal Data Flows Through PE



- Active packets move through configured kernel plugins.
  - » active function dispatcher passes packets to instances of plugin objects
  - » instantiates objects or triggers download of plugin class, as needed
  - » streamlined processing of SAPF packets using pre-established state
- Standard processing for “plain” IP packets.
  - » classification and routing, header processing, output queueing

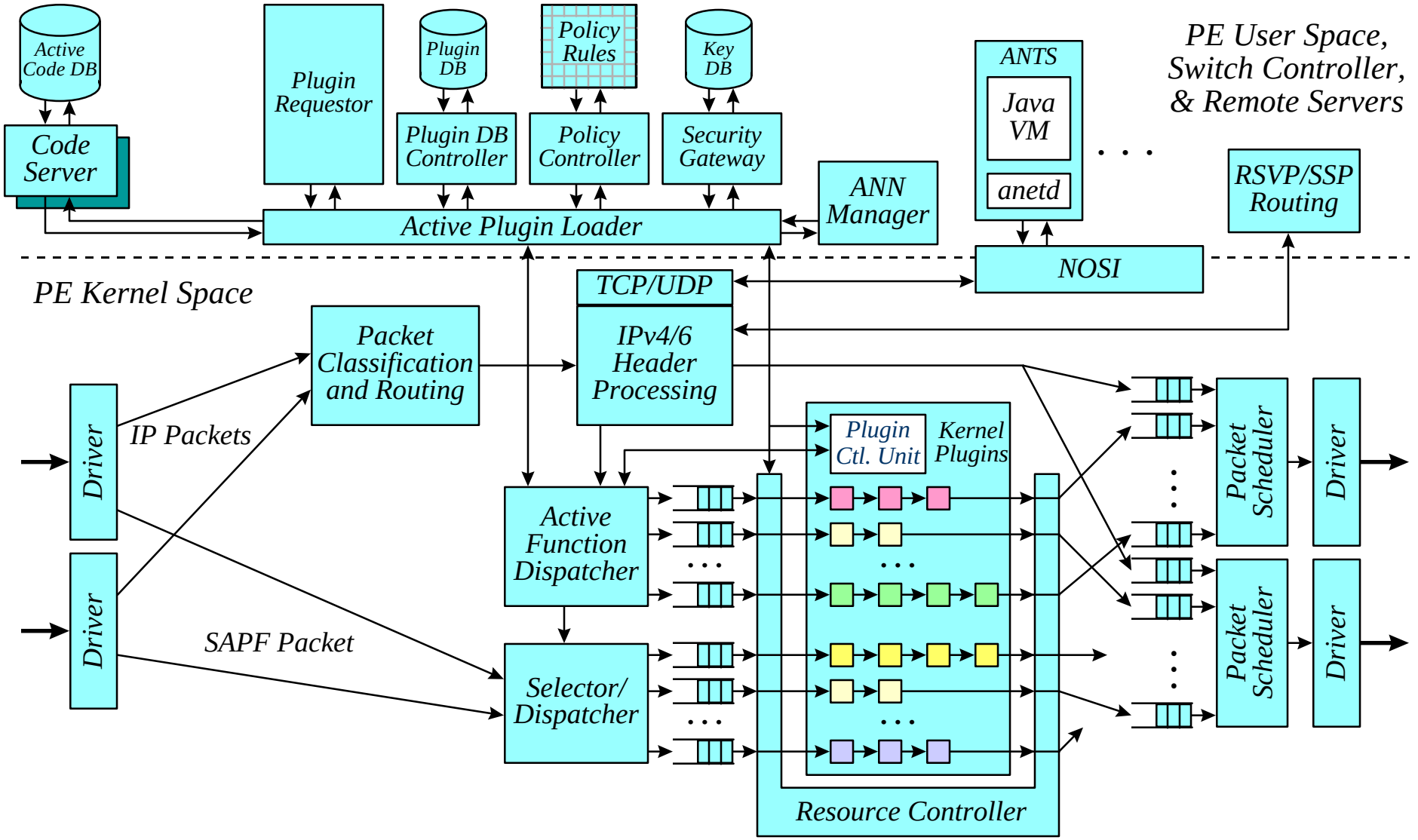
Overall System Architecture

- *Switch Controller* (SC) manages system.
  - » global coordination & control
  - » routing protocols
  - » build routing tables and other information needed by PEs
  - » first level code server
  - » reprogrammable for active processing



- Processing Elements (PE) do per packet processing.
  - » standard processing of IP packets
  - » various types of active processing
    - – Active IP packets using Active Network Encapsulation Protocol (ANEP)
    - – Simple Active Packet Format (SAPF) using pre-established flow state (conceptually similar to tag switching)
    - – ANTS for capsule execution in user-space Java VM
  - » control mechanisms for demand-loading of plugins

Software Architecture



Active Packet Processing Components

Active Function Dispatcher

- parses packets of new active flows
- determines what plugins need to be loaded and instantiated

Resource Controller

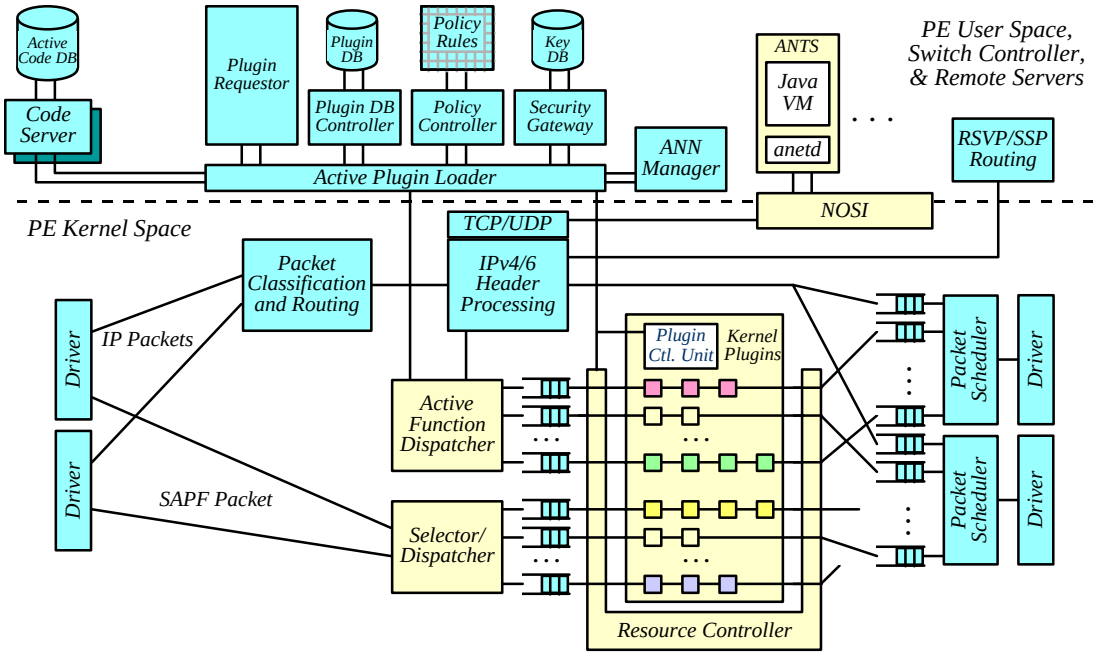
- track CPU and memory usage by active flows
- enforce usage limits
- DRR-style algorithm

Plugin Control Unit

- maintain resident active plugins
- support plugin loading / unloading

Selector/Dispatcher

- implements the Simple Active Packet Format
- enables more efficient packet classification for active flows



ANTS

- user space execution environment developed by MIT

NodeOS API

- standard AN interface on which for user space EEs
- provides abstractions for computation, storage and communication.

Plugin Management

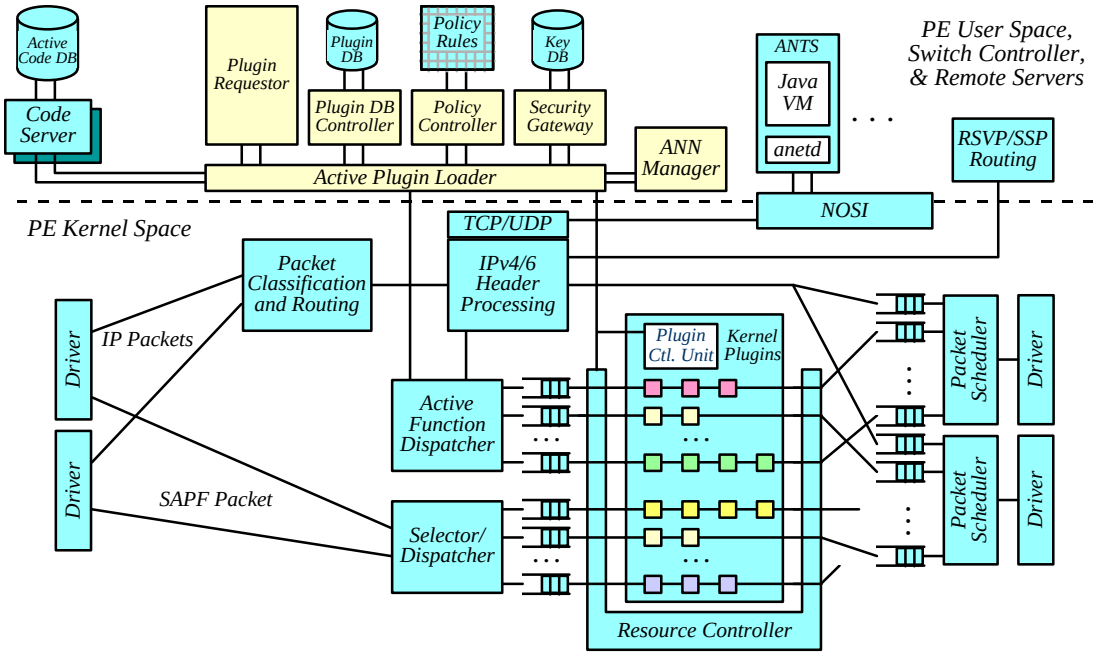
Plugin Requestor
- communicates with code servers to retrieve new plugins

Active Plugin Loader
- provides flow of control among the other plugin management components

Plugin Database Controller
- administers local plugin database (cache)

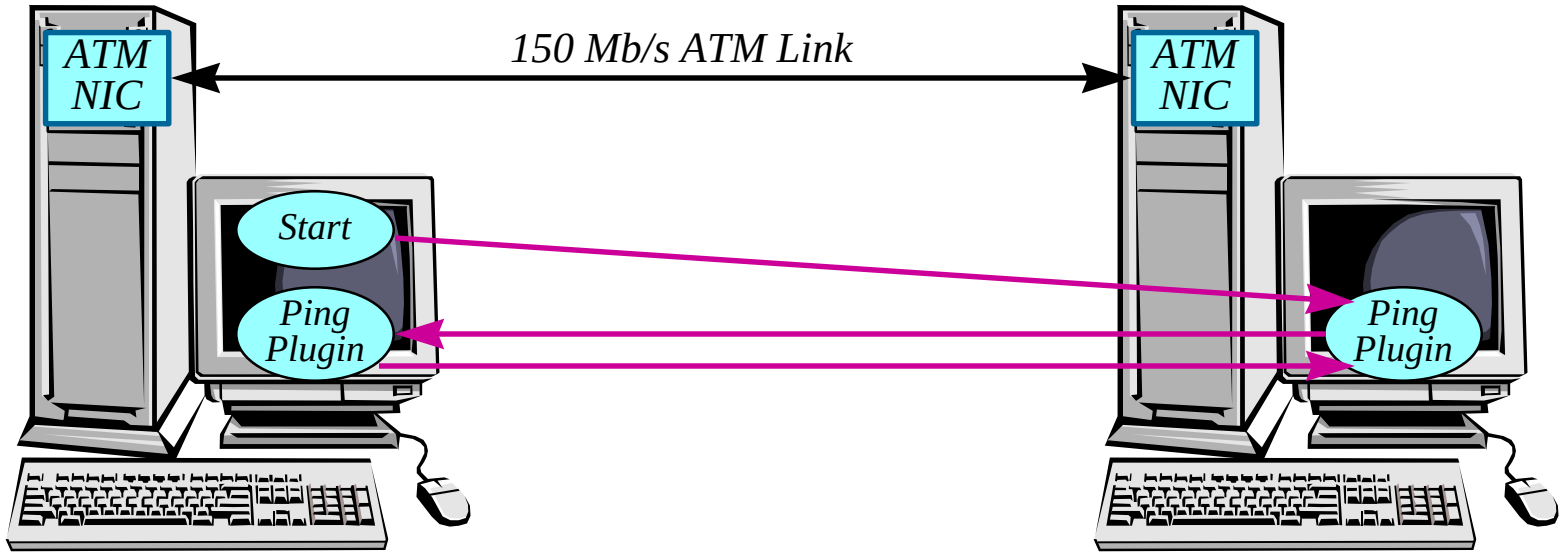
Policy Controller
- controls admission of new plugins based on policy rules
- maintains policy database

Security Gateway
- checks integrity and origin of new plugins
- maintains database (cache) of public keys for developers, code servers.



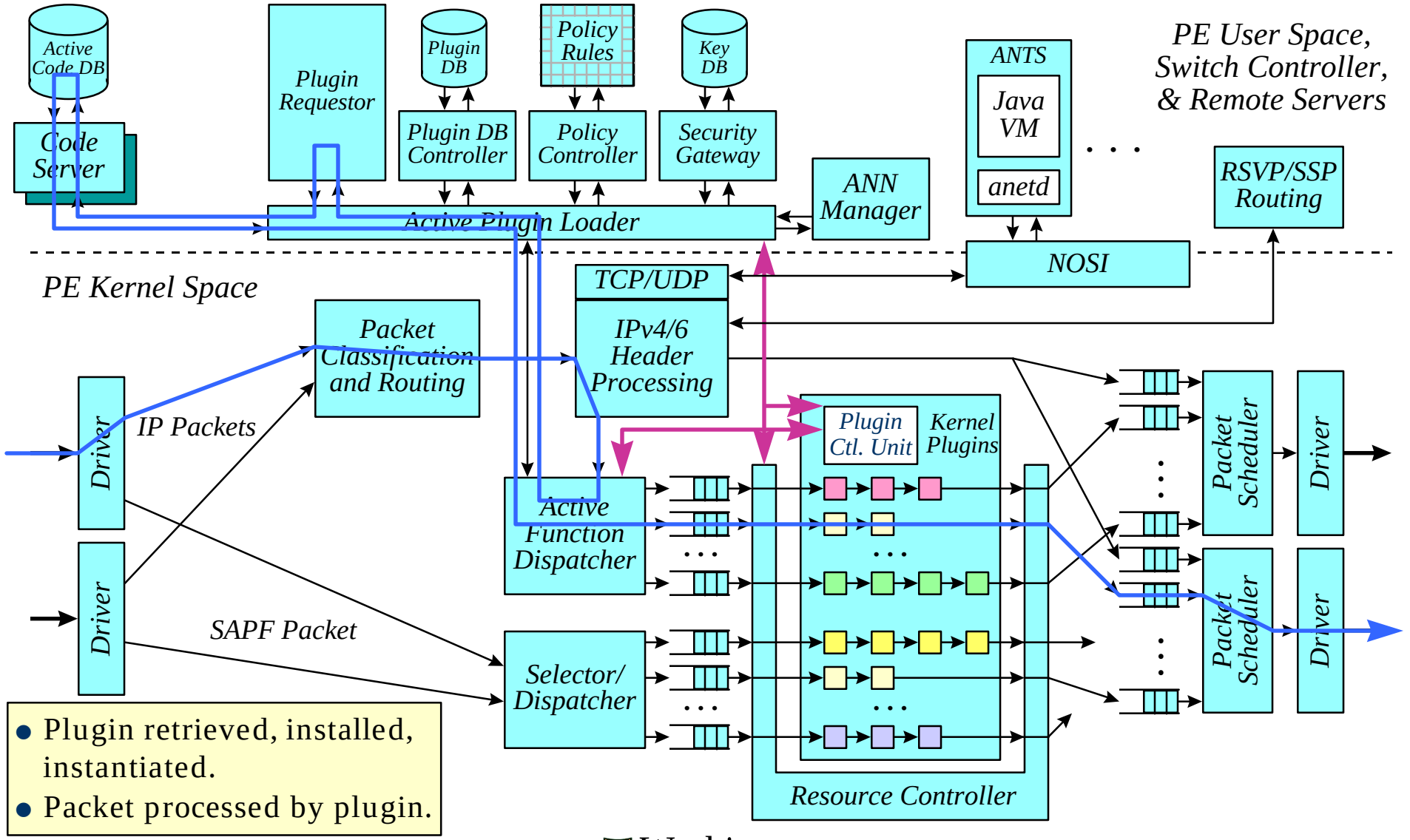
AN Node Manager
- user interface for node management
- supports privileged access to other software components

Basic Demo & Performance Measurement



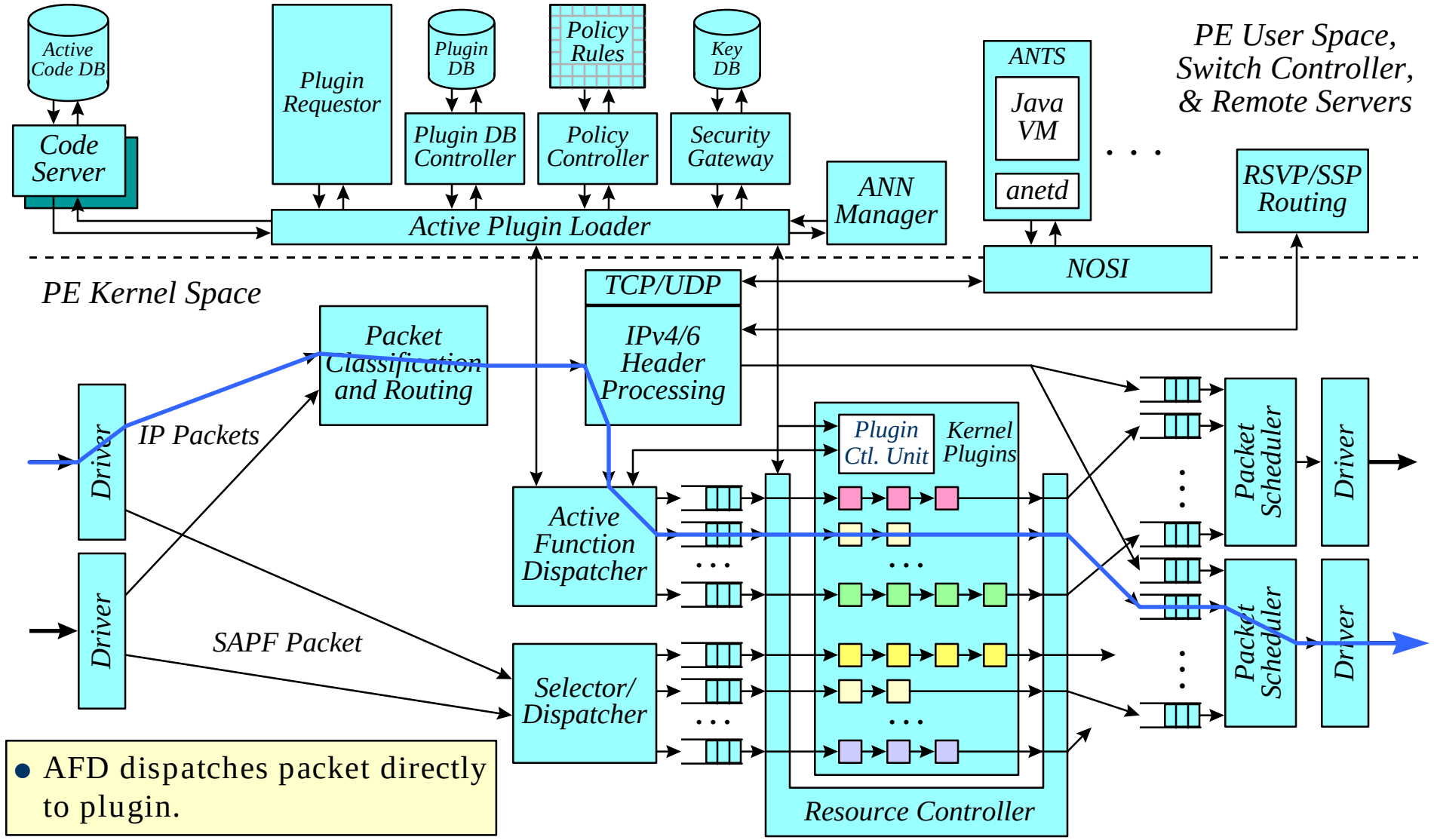
- User-level program (start) sends active packet referencing ping plugin to remote machine.
- Ping plugin swaps source / destination addresses and forwards packet.
- After specified number of round trips, packet returned to user level program to generate measurement.

Ping (first packet)



- Plugin retrieved, installed, instantiated.
- Packet processed by plugin.

Ping (second packet)

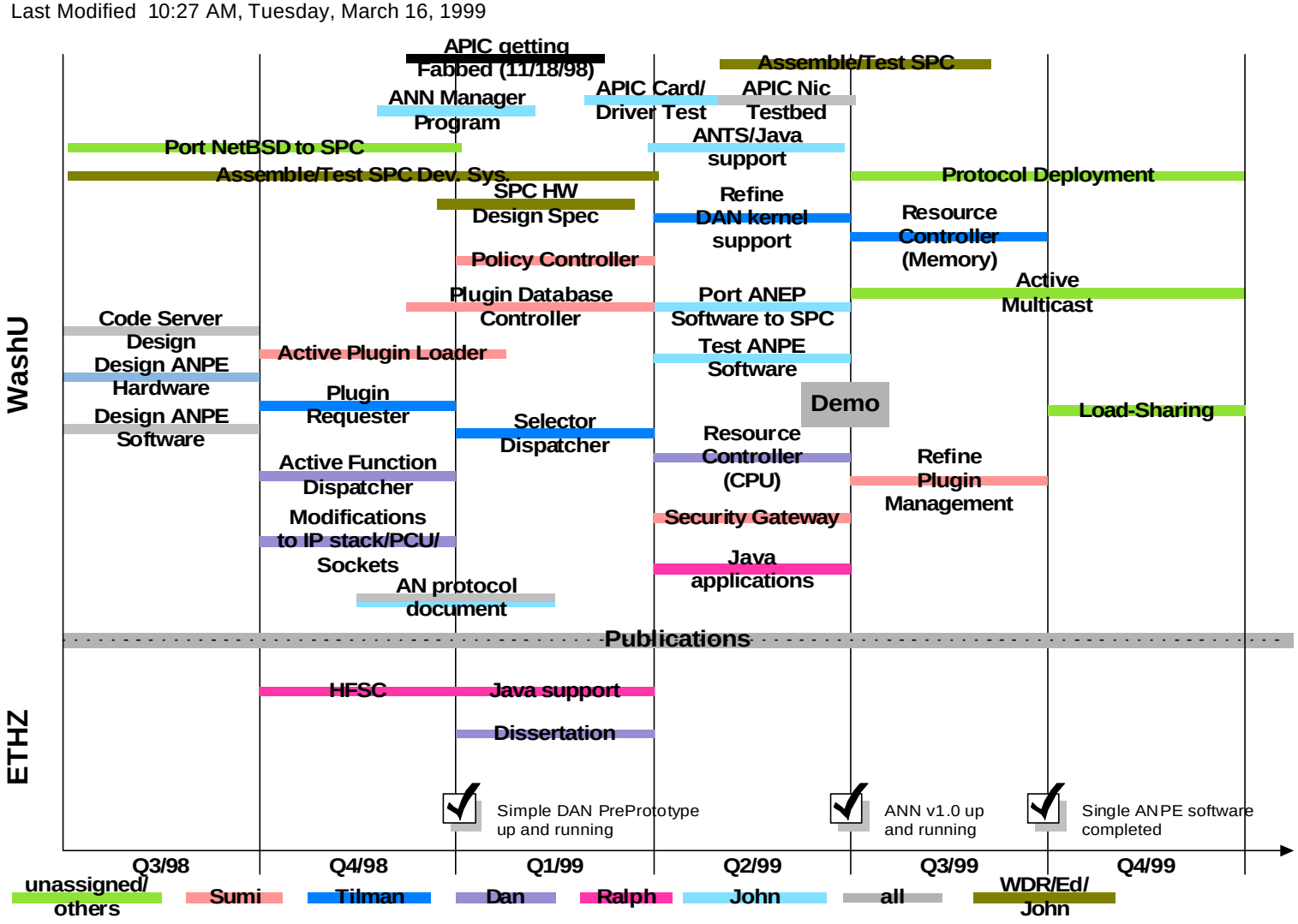


- AFD dispatches packet directly to plugin.

Ping Performance

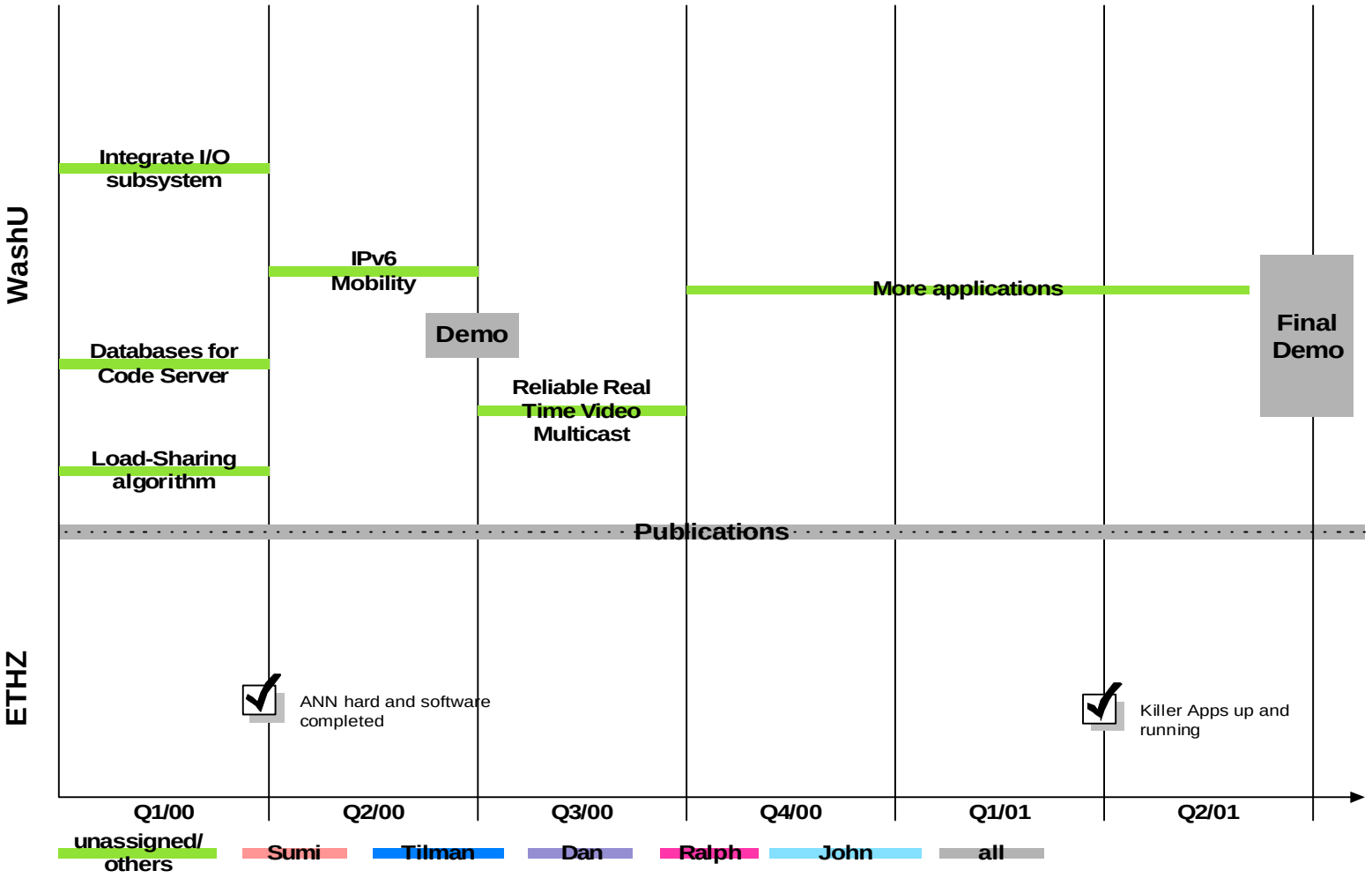
- Experimental Setup
  - 233 MHz Pentium Pro
  - Efficient Networks ATM interface (150 Mb/s)
- Results (after first packet)
  - round trip delay 169 µsec
  - one way delay 84.5 µsec
  - shortcut forwarding 50 µsec (hardware 40, software 10)
  - AFD & plugin 34.5 µsec
- Comparisons
  - U Penn PLAN (PII 300 MHz) 200 µsecs
  - MIT, ANTS (UltraSparc 167 MHz) 588 µsecs
  - Arizona, Scout (200 MHz) 518 µsecs

ANN Project Schedule (page 1)

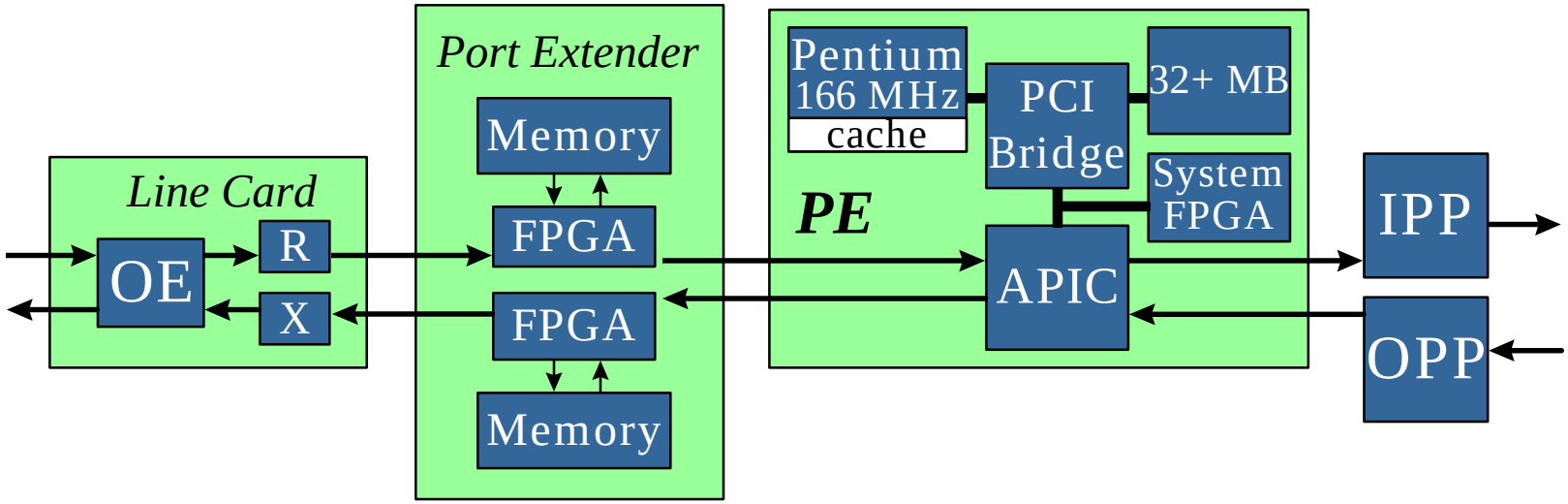


ANN Project Schedule (page 2)

Last Modified 10:29 AM, Tuesday, March 16, 1999

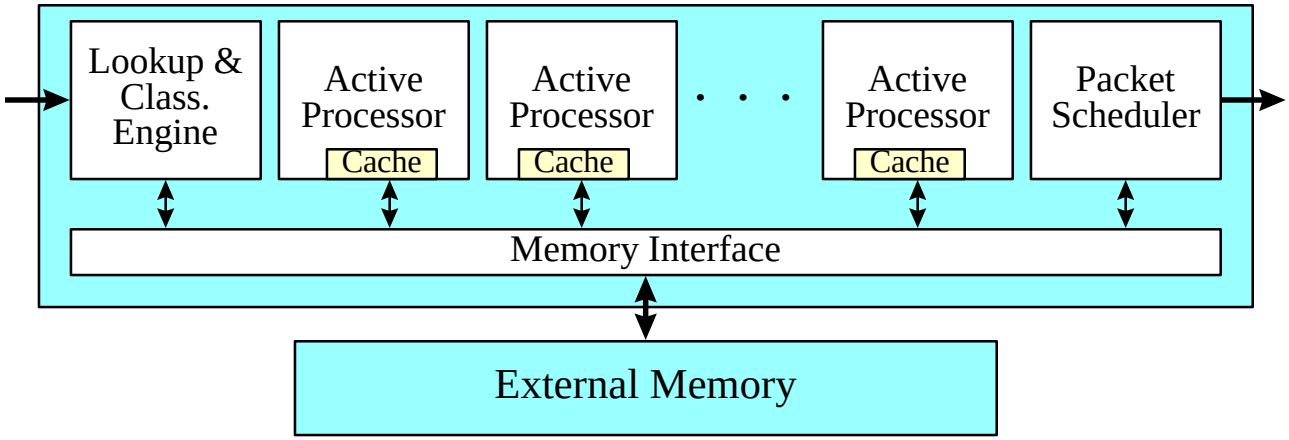


Configurable "Active Hardware"



- Port Extender can be stacked with PE at each port
- On-board FPGAs can provide support for active processing.
  - » fast packet classification and "plain" IP forwarding
  - » output queue scheduling
- FPGAs can potentially be "programmed" for active processing
  - » "hardware plugins" for computationally intensive active applications

Towards Commodity Active Processing



- Large-scale success of active networks will require to single chips capable of active processing at 1-10 Gb/s.
- Technology trends favor multiple processors on a chip.
  - » single chip processor performance scales sublinearly with chip area
  - » multiprocessor performance scales linearly
- Design trade-offs different than with commercial single processor chips.

Commercial Processors (desktop)

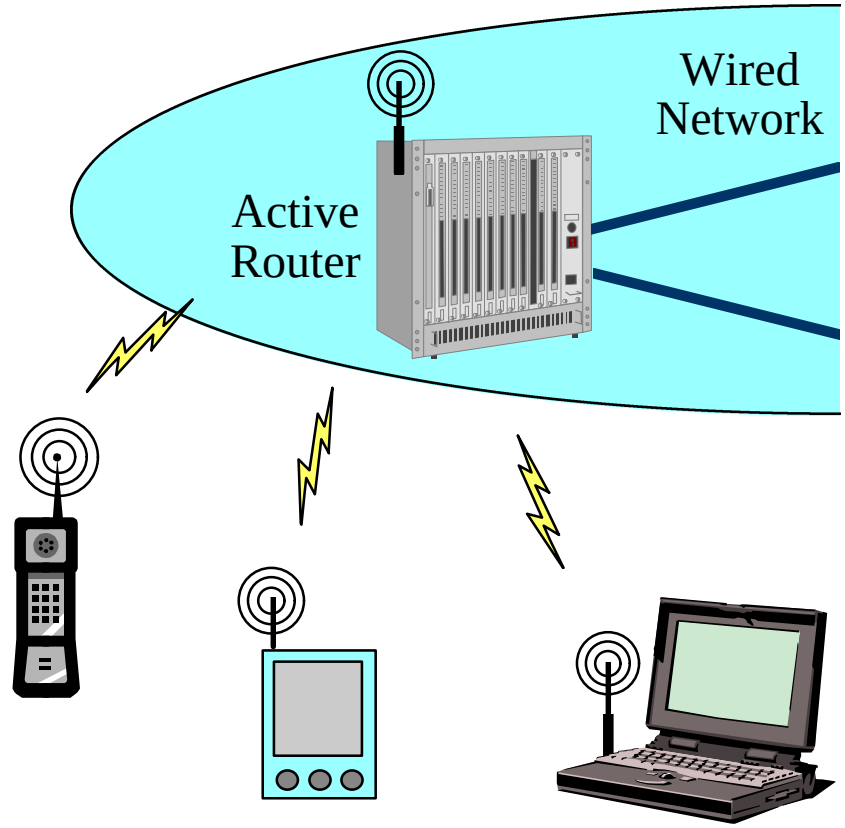
- » few concurrent tasks
- » virtual memory, floating point important
- » coarse-grained processor scheduling is adequate

Active Processors

- » many concurrent tasks
- » VM, FP much less important
- » finer grained scheduling needed

Active Processing for Wireless Communication

- Constraints on computational power of wireless devices.
  - » power, cost, weight
- Active router can act as computational agent for wireless devices.
  - » wireless communication triggers activation of device-specific active program
  - » may be downloaded from remote server
  - » enables rapid bug fixes, upgrades and user-specific capabilities
- Generalizes to wireless hierarchy.
  - » portable hand-held devices at base of hierarchy
  - » vehicle-mounted routers provide relay to satellites and wired network infrastructure



Conclusions

- High performance Active Networking need not be an oxymoron.
  - » scalable systems with gigabit links and terabit throughputs are possible with current / near-term technology
  - » on-going technology improvements will make AN economically viable
- Our approach sacrifices some flexibility relative to capsules.
  - » sacrifice seems appropriate for performance gains
  - » since same platform can also support capsules, no real loss of flexibility
- More work needed on active applications
  - » need some compelling instances of active applications to focus design efforts
  - » benchmarks needed for performance evaluation
  - » need better understanding of "API" for active code developers
- More information:

http://www.arl.wustl.edu/arl/projects/ann/ann.html