Open Networking
Better Networking Through Programmability

Extensible Router Workshop
Princeton University

Tal Lavian
Nortel Networks Labs
tlavian@ieee.org

More info: http://www.openetlab.org

Usual Disclaimer

We are part of research organization

This talk describes exploratory research

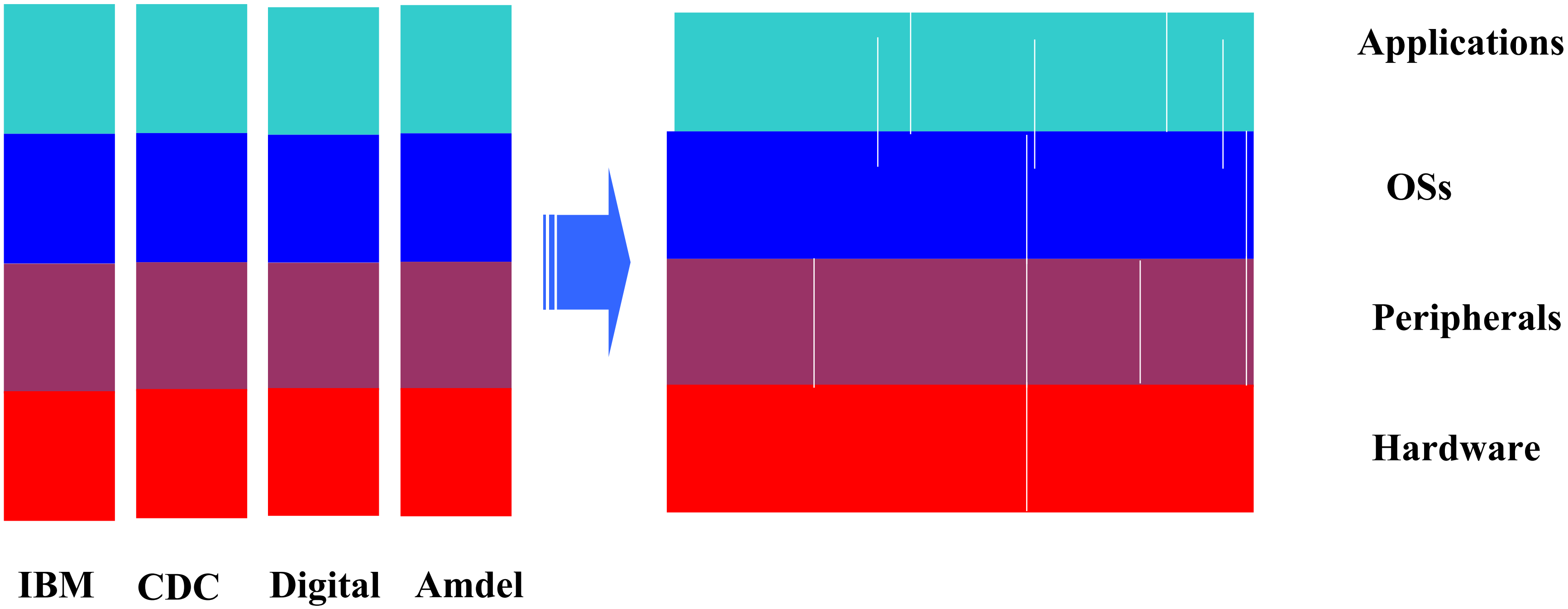
- No commitment by Nortel to turn technology into product.
- No commitment by Nortel to do anything with ideas described in this talk.

Programmable Network Devices

Openly Programmable Devices Enable

New Types of Intelligence on the Network

Industry movement from vertical toward horizontal markets



Applications

OSs

Peripherals

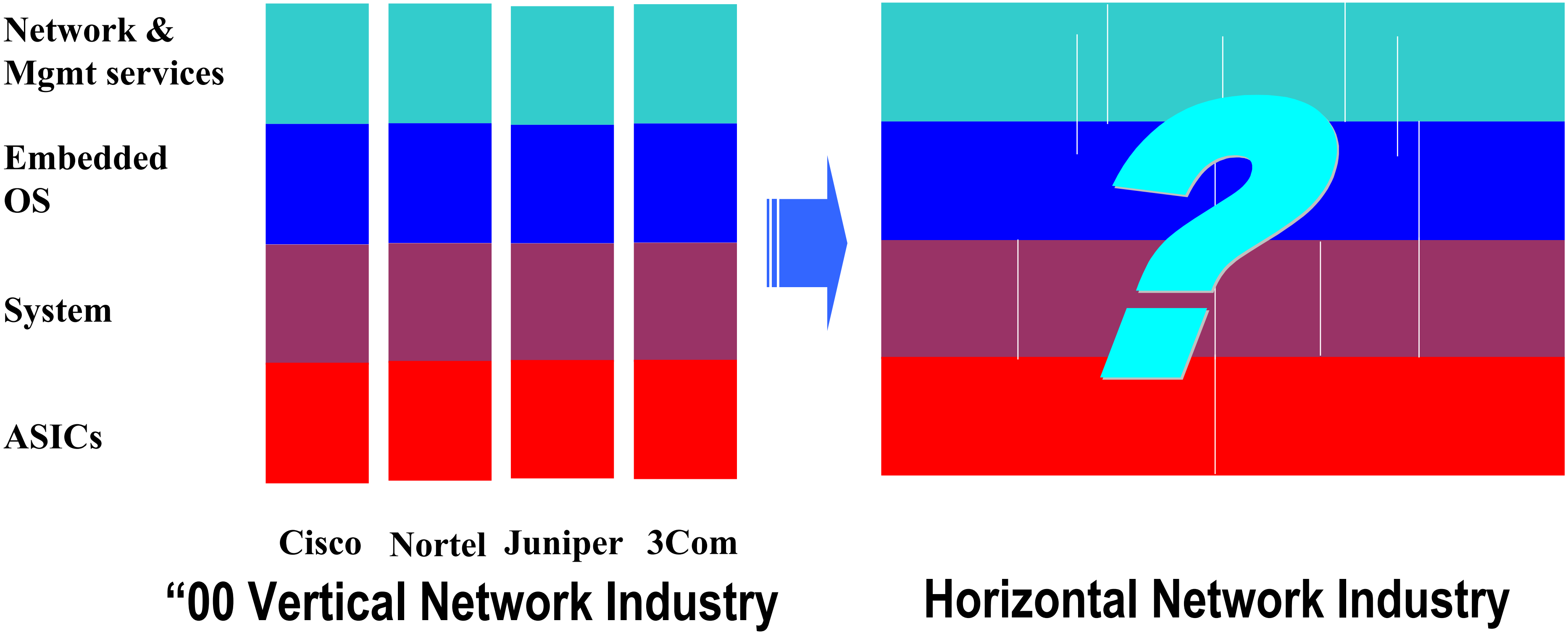
Hardware

IBM CDC Digital Amdel

1980s - Vertical Industry

2000s - Horizontal Industry

Inflection Points Ahead of Us



Incomplete transformation; the inflection point is quickly approaching …

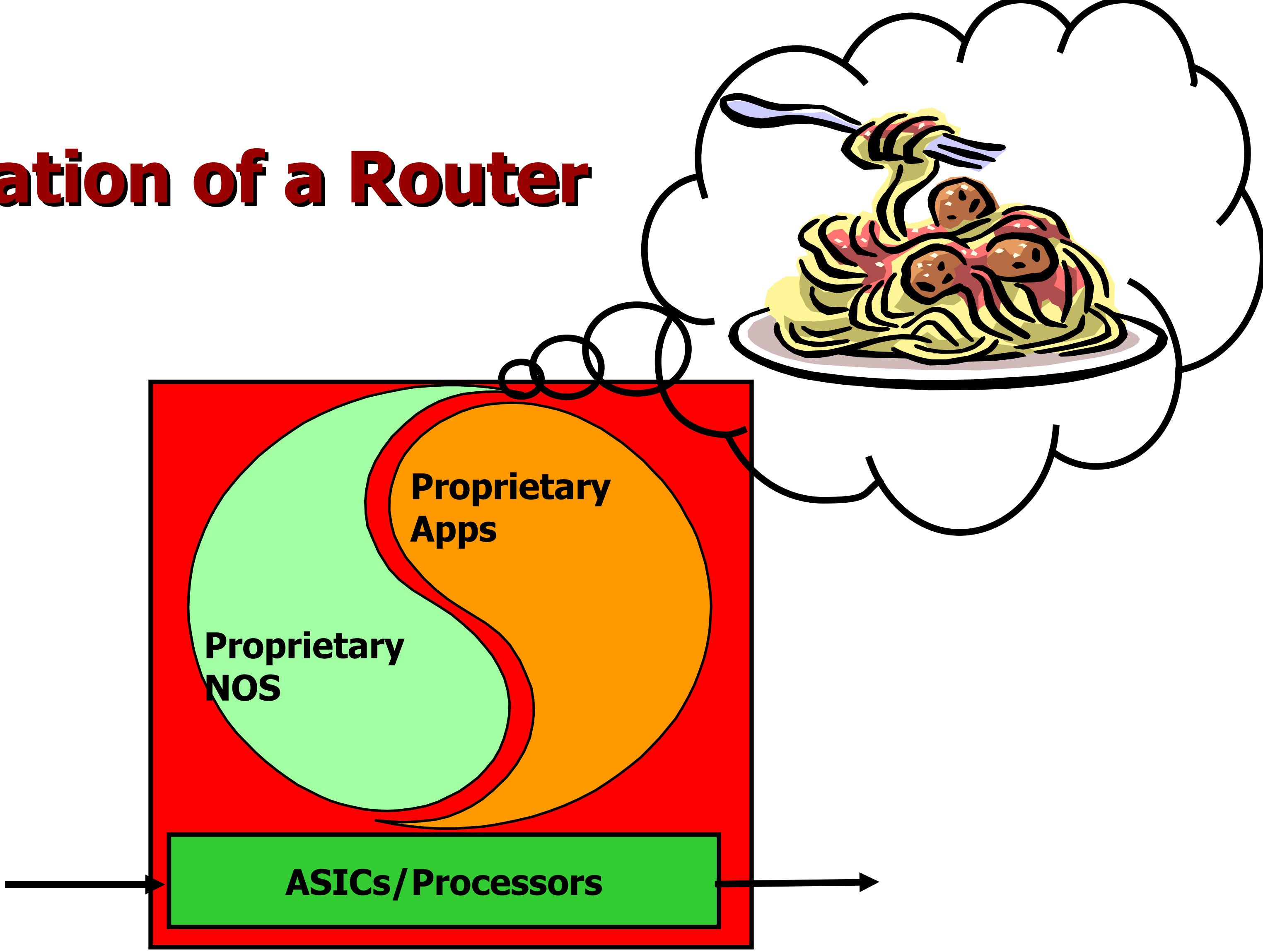
Location, location, location

Service-enablement will prove most effective where "impedance mismatches" occur in the network

- Optical vs. Wireline
- Wireline vs. wire-less
- Secure vs. non-secure
- Customer-premises vs. Content-provider-land
- SLA (x) vs. SLA (y)
- Resource-constrained vs. unwashed unlimited computing

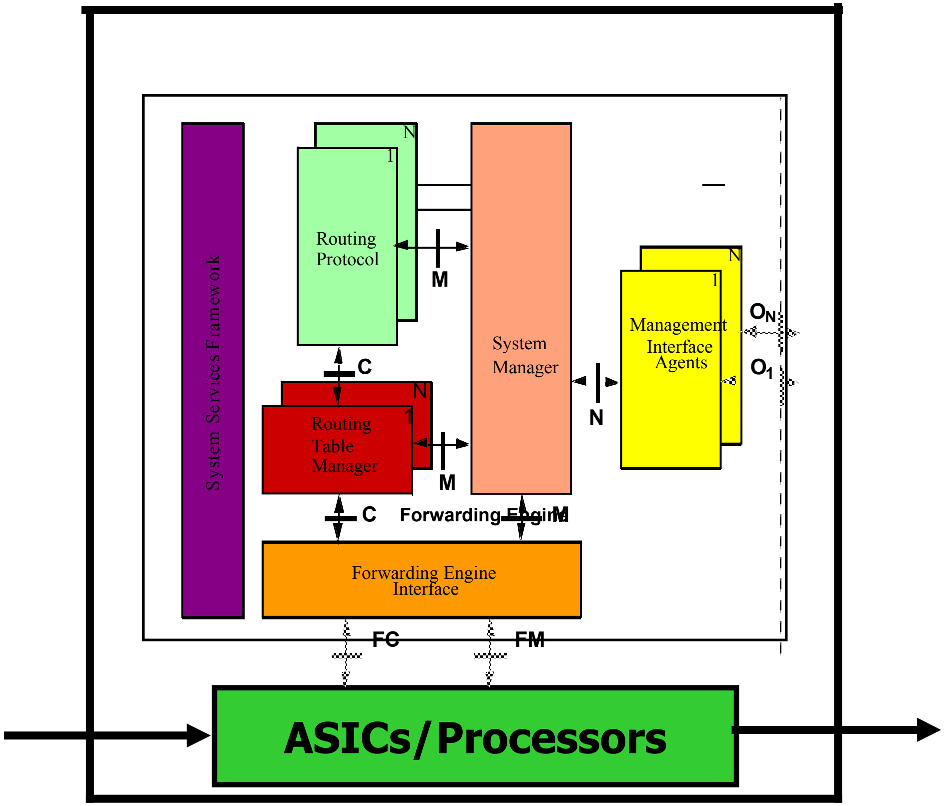
A service-enabled box can wear multiple hats

Emancipation of a Router



It all started from old-world, vertically-integrated code.

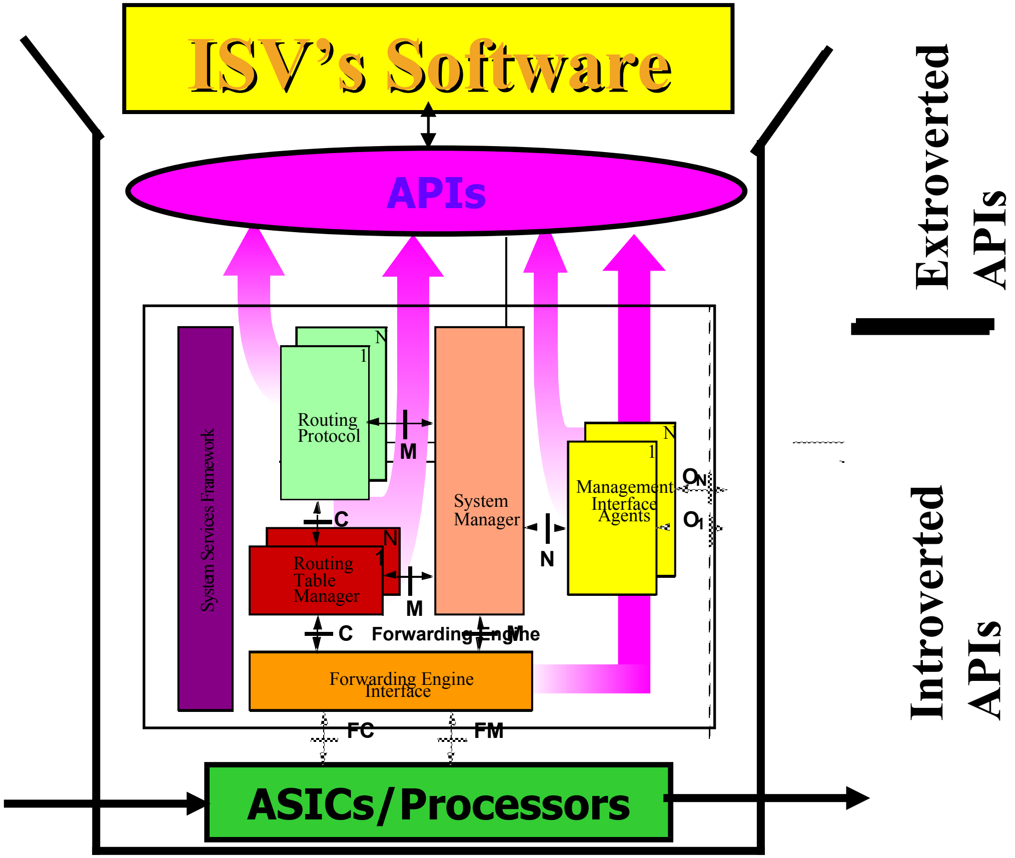
1st Degree of Emancipation



Introverted APIs Emerge.

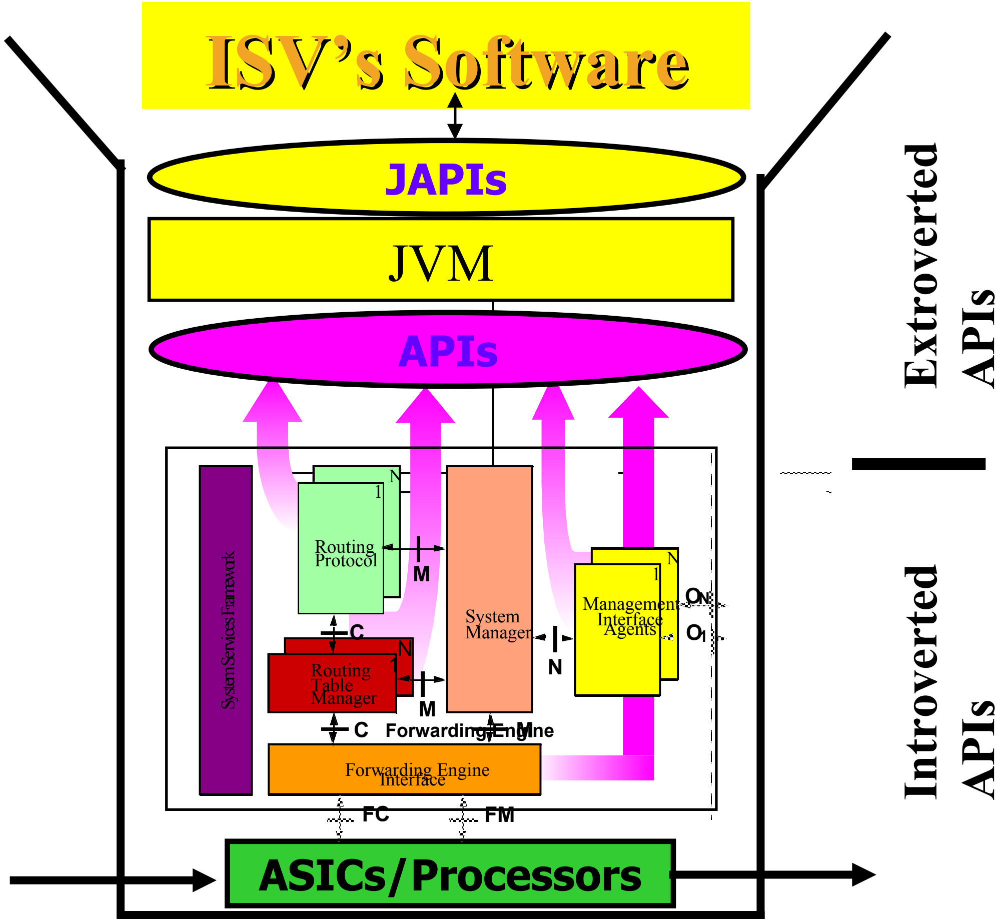
Modular code is native, local, and trusted. port required.

2nd Degree of Emancipation



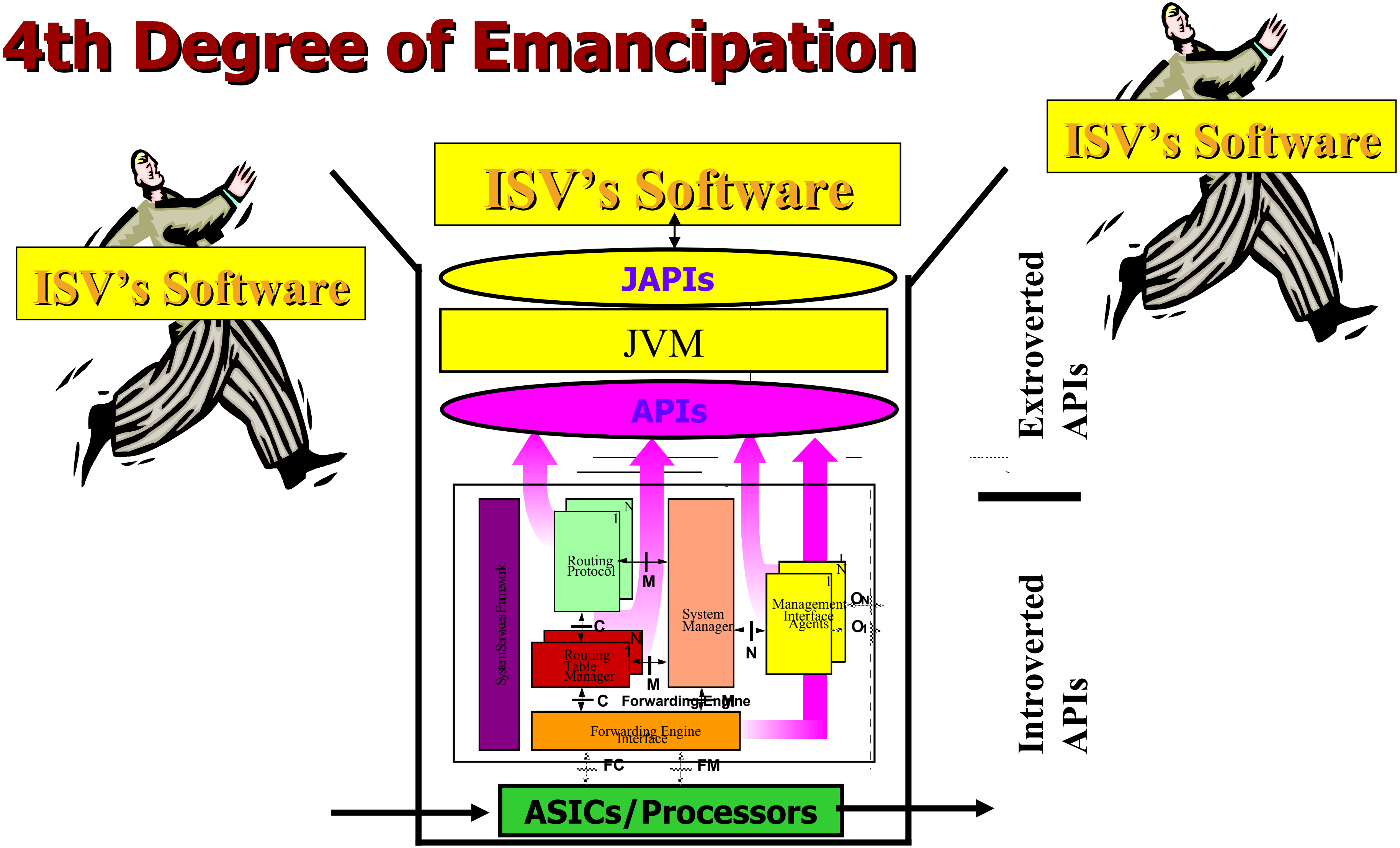
Extroverted APIs expose object capabilities to ISV code.

3th Degree of Emancipation



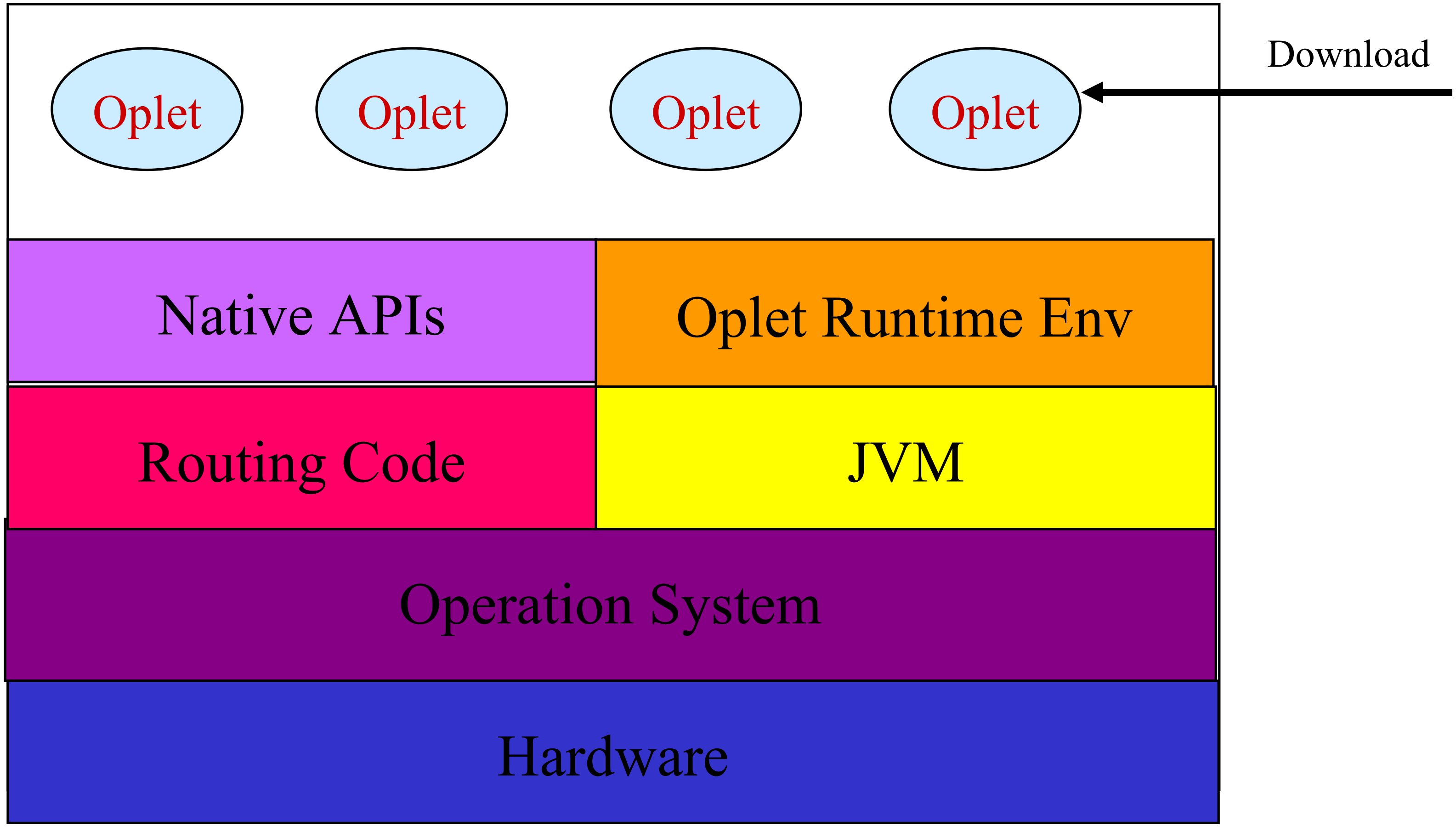
Extroverted APIs extend a commodity Java runtime.

4th Degree of Emancipation

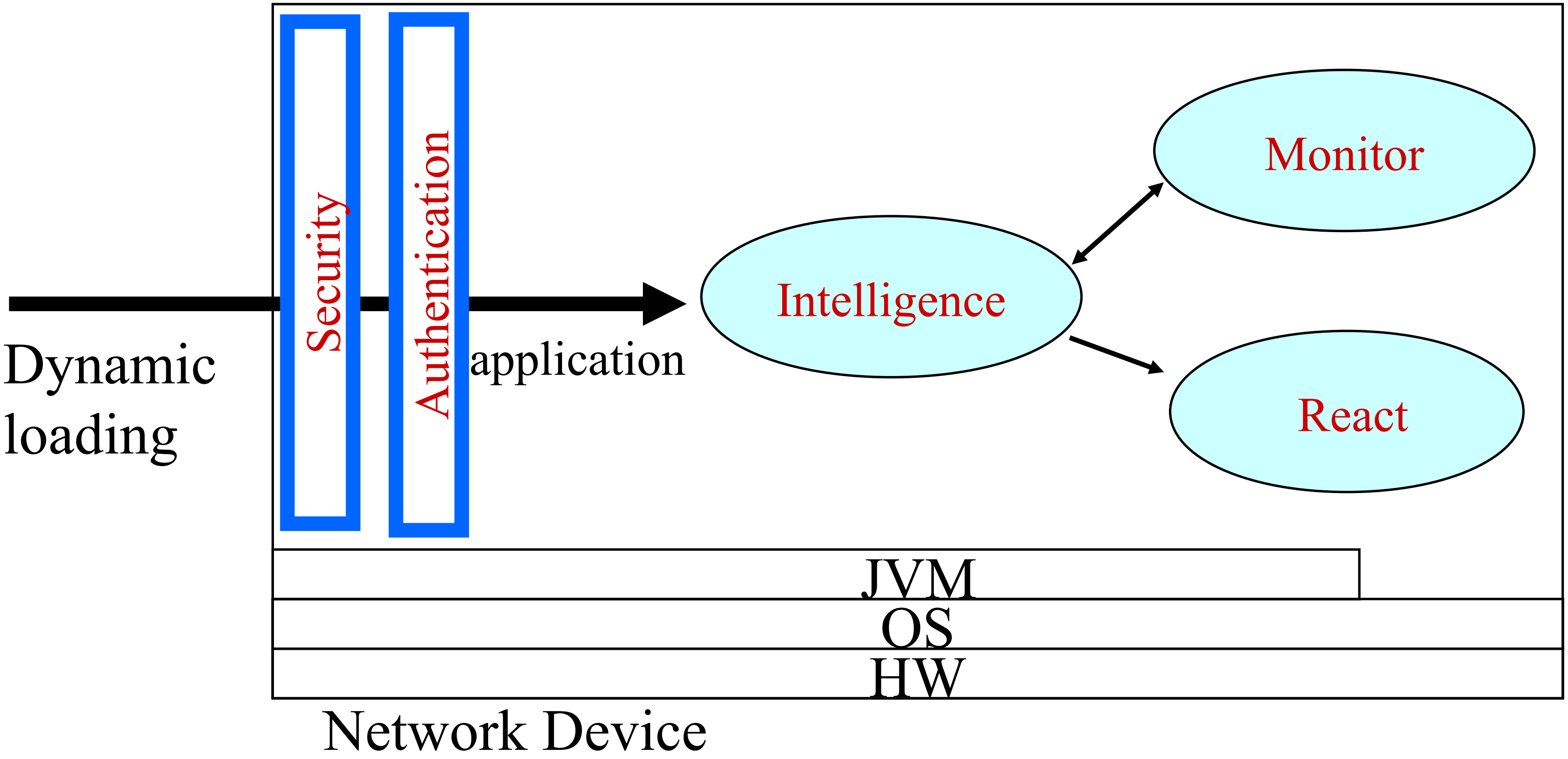


ISV code is local/non-local, non-native, non-trusted, loaded on demand, and can teleport itself.

Java-enabled Device Architecture



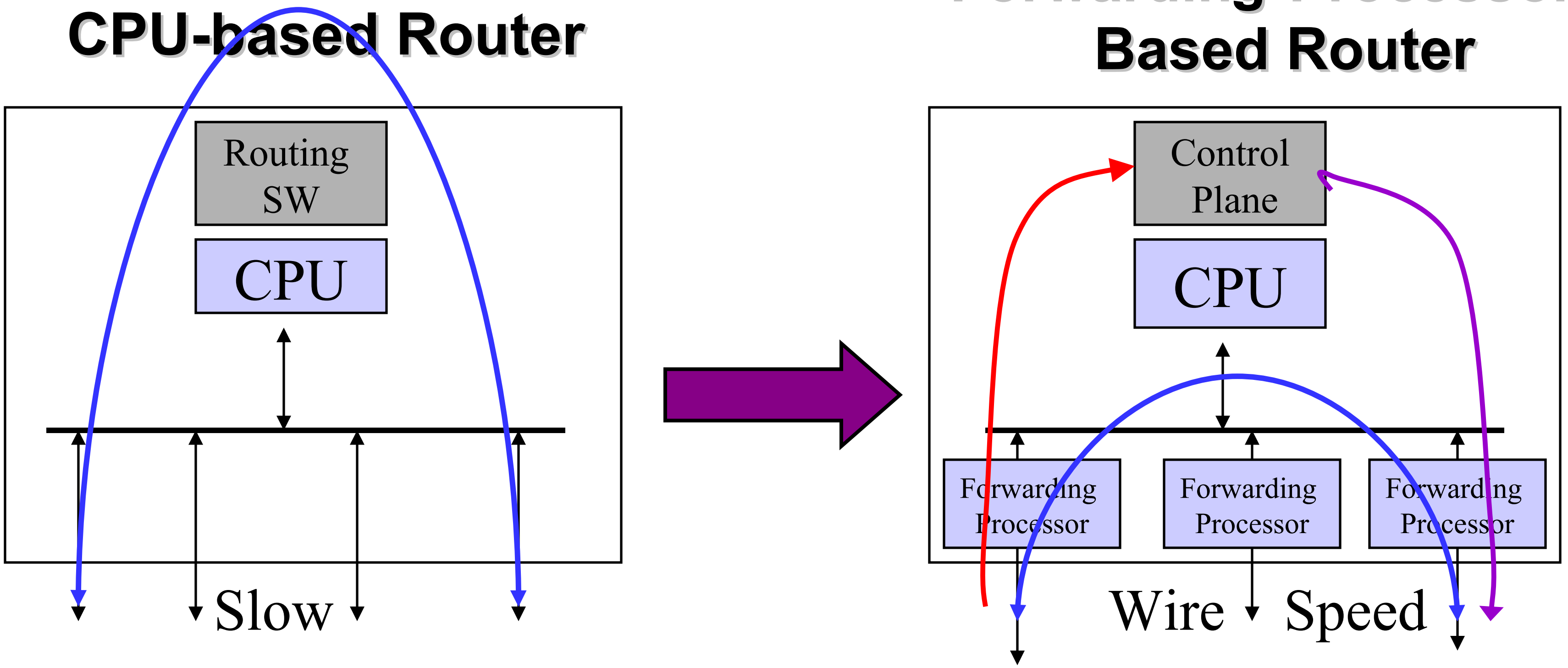
Example: Downloading Intelligence



Separation of Control and Forwarding Planes

Centralized, CPU-based Router

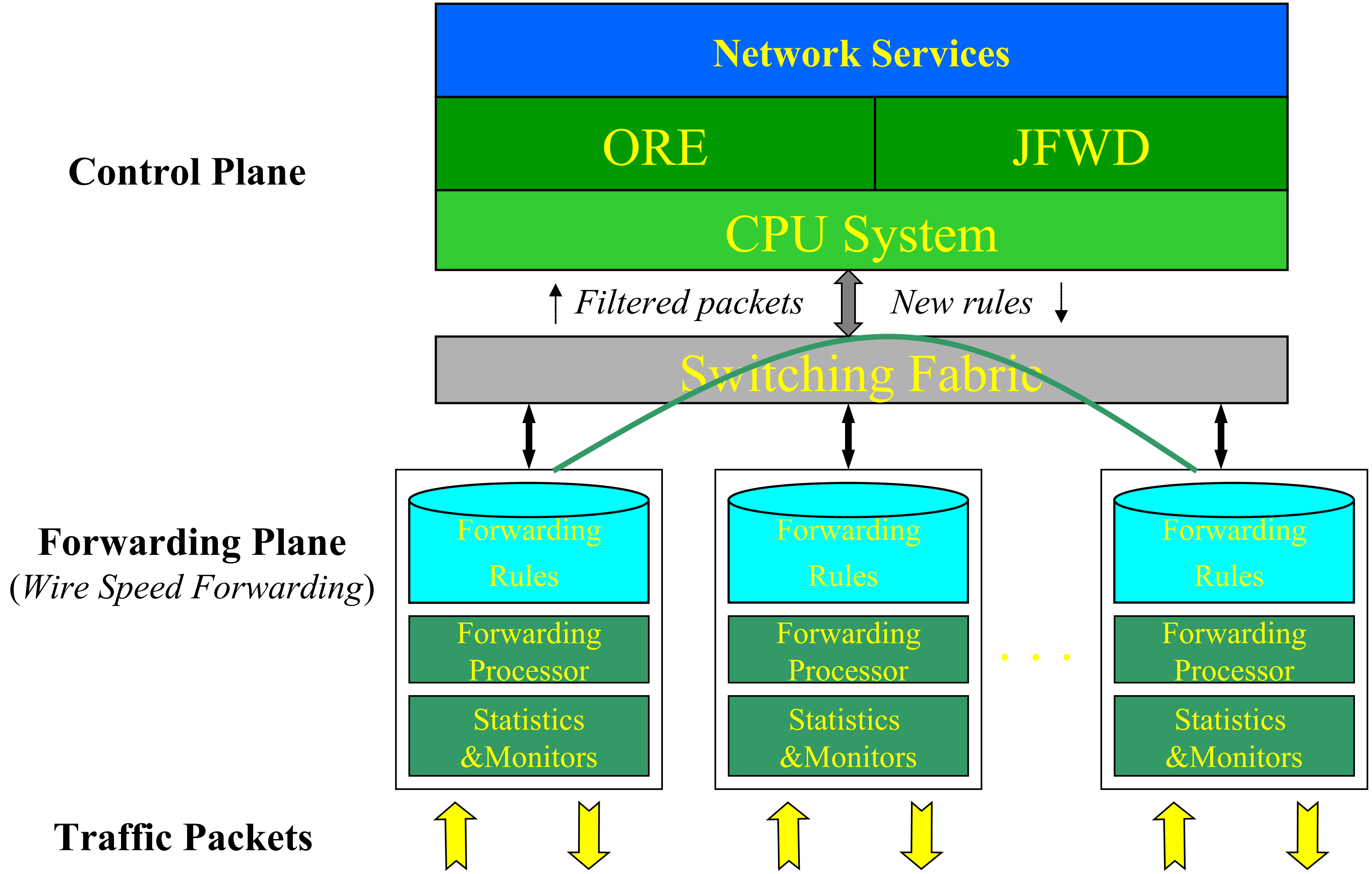
Forwarding-Processors Based Router



Control + Forwarding Functions combined

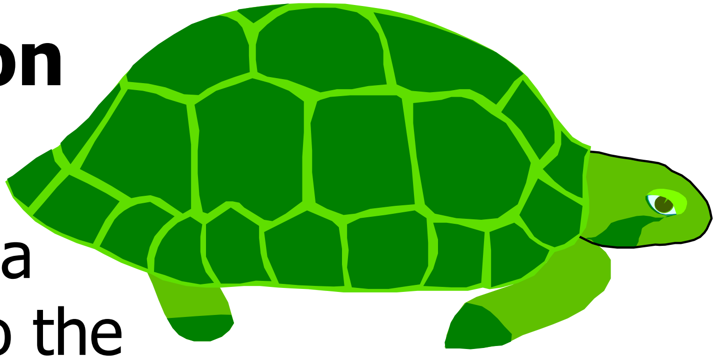
Control separated from forwarding

Programmable Networking



But Java is Slooowwwww

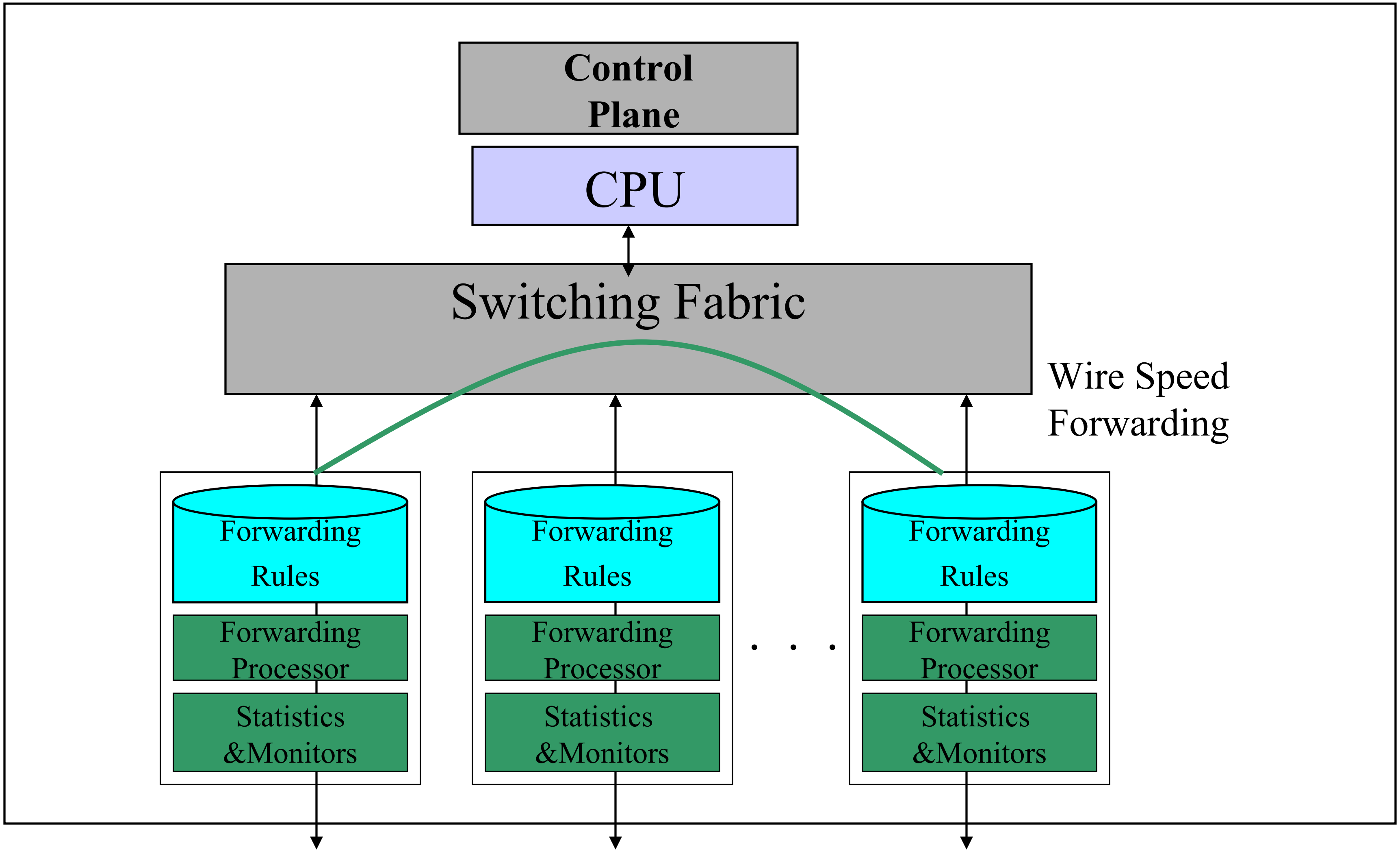
- **Not appropriate in the fast-path data forwarding plane**
  - forwarding is done by ASICs
  - packet processing not affected
- **Java applications run on the CPU**
  - Packets designated for Java application are pushed into the control plane



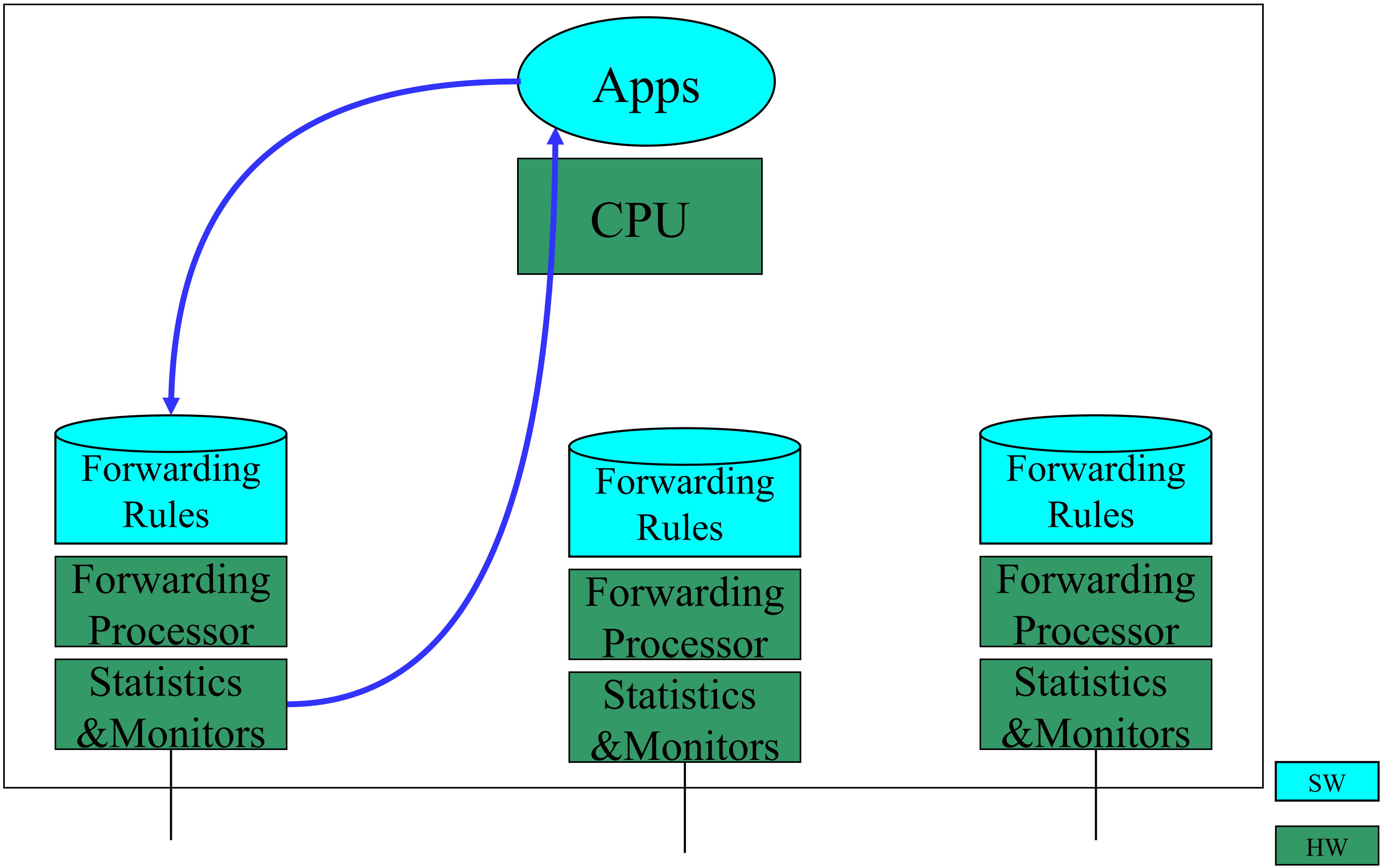
Simple Example: Fine grain monitoring

- **Imagine a SNMP-based network with:**
  - 100 nodes
  - each node with 100 ports
  - each port with 100 conditions
  - all being checked 100 times a second
- **That's 10 billion SNMP variable accesses every second.**
- **And that's a significant load on the NMS and the network as a whole. It's not going to work.**

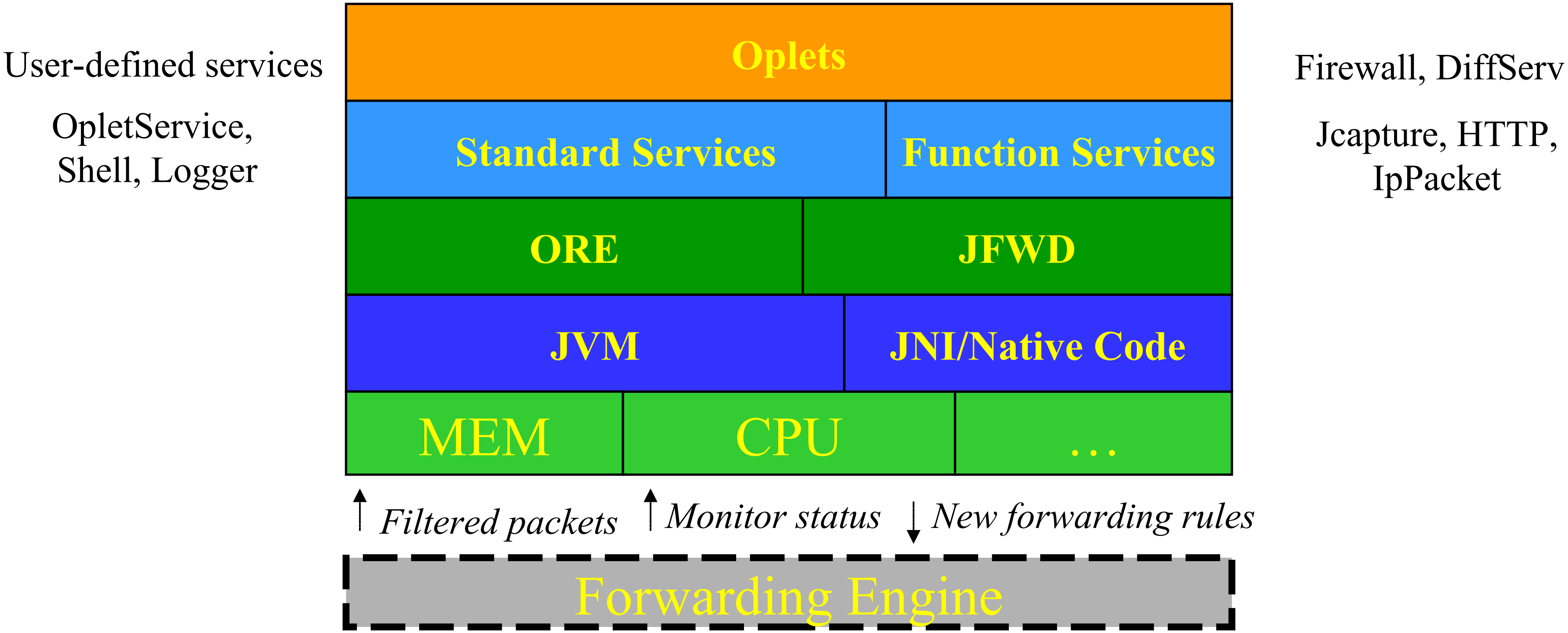
Silicon-based Forwarding Engines



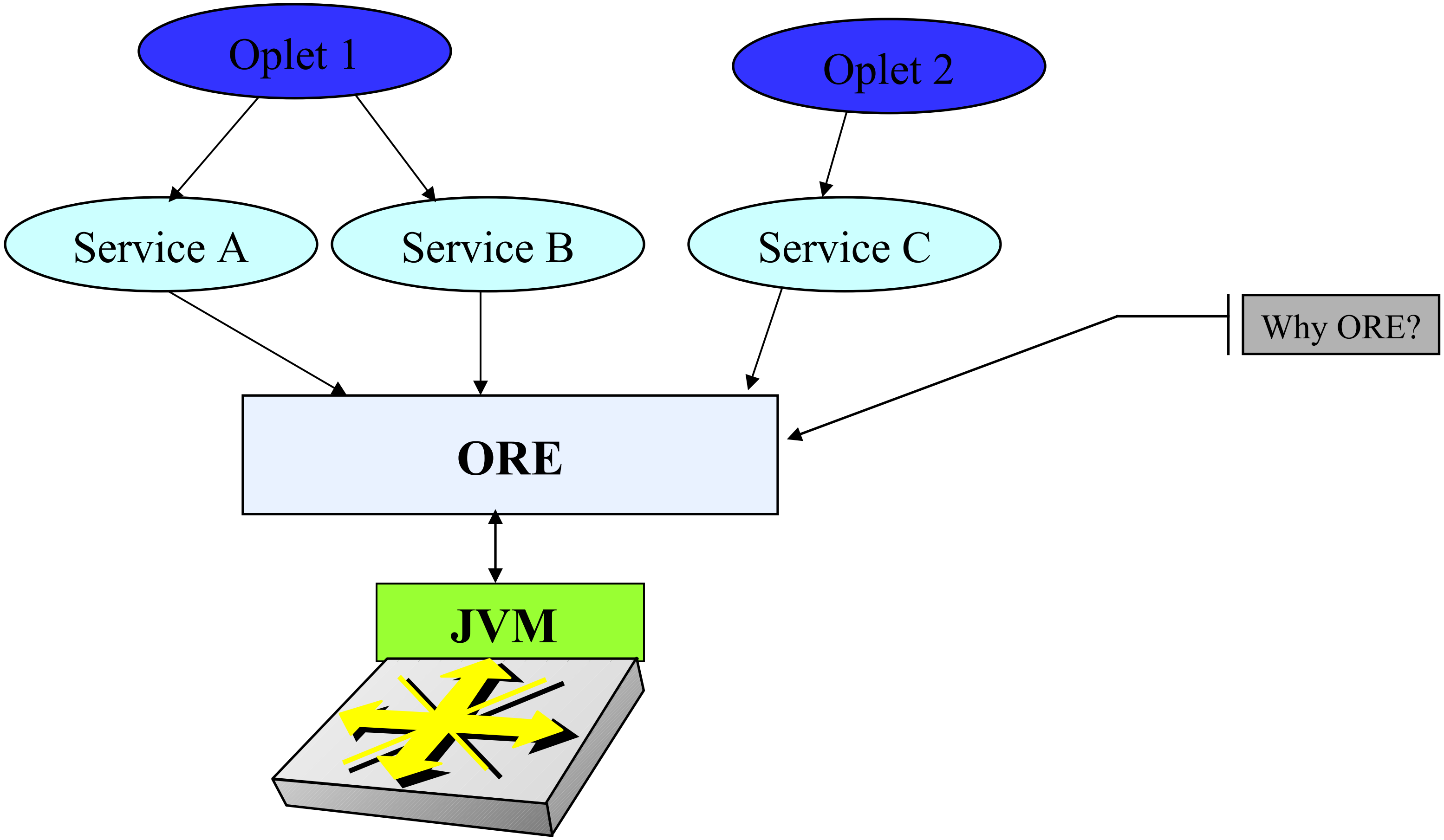
Real-time Forwarding Stats and Monitors



ORE Architecture



ORE - Oplet Run-time Environment



Basic ORE Concepts

- **Oplet Runtime Environment (ORE)**
  - A kernel that manages the life cycle of oplets and services
  - Provides a registry of services
- **Services**
  - The value being added. Minimal constraint, could be anything
  - Represented as a Java interface
- **Oplets**
  - The unit of deployment: a JAR file
  - Contains meta-data (e.g. signatures, dependency declarations)
  - Contains services and other resources (data files, images, properties, JAR files)

Oplet Lifecycle

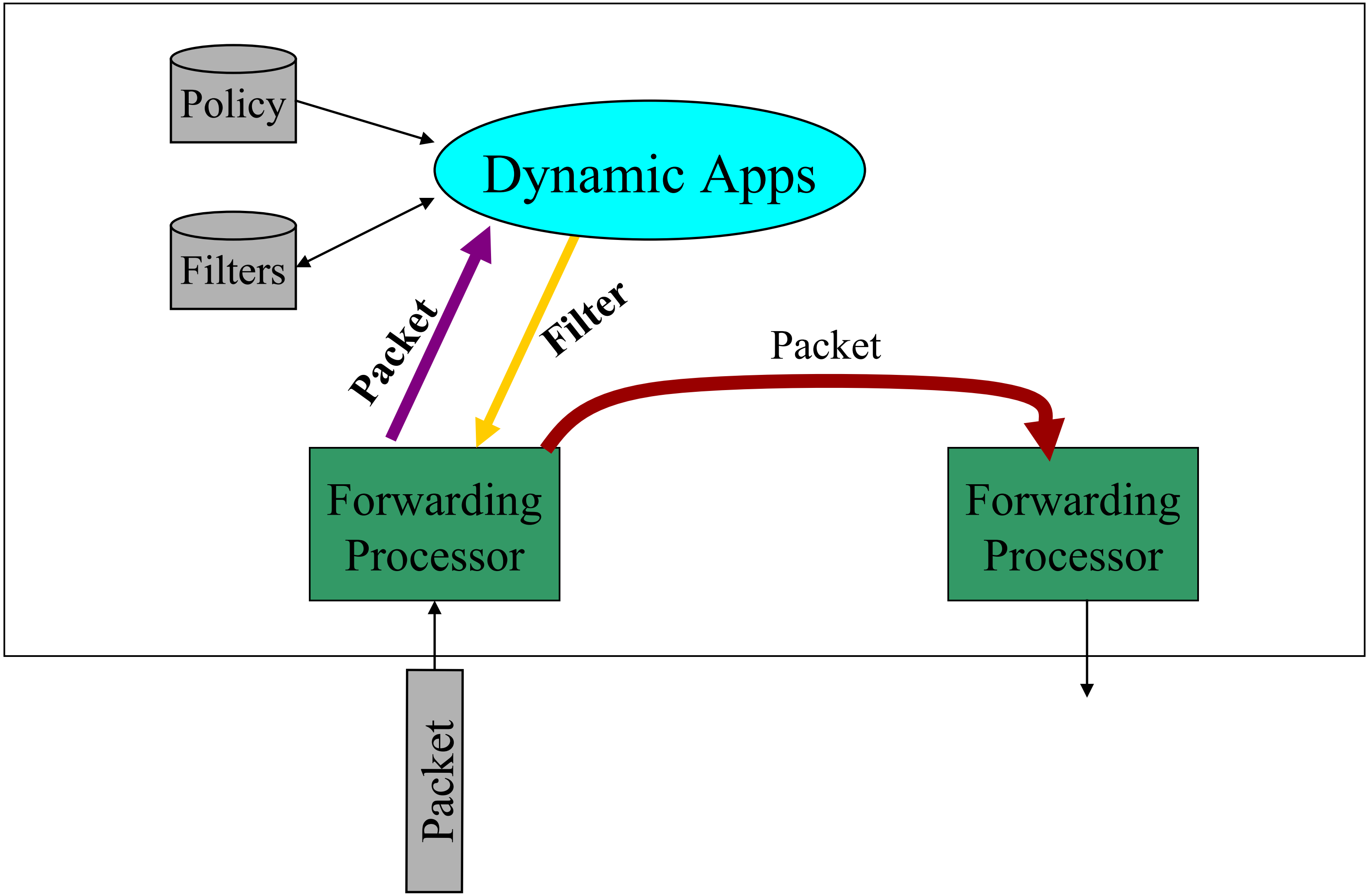
- **Install**
  - Loaded from URL
- **Start**
  - Services that are depended on must already be started
- **Stop**
  - Any oplets that depend on this oplet’s services will be stopped
  - Code and data can be unloaded from ORE
- **Update**
  - Updates the oplet with a new oplet
- **Uninstall**

Dynamic Classification Objectives

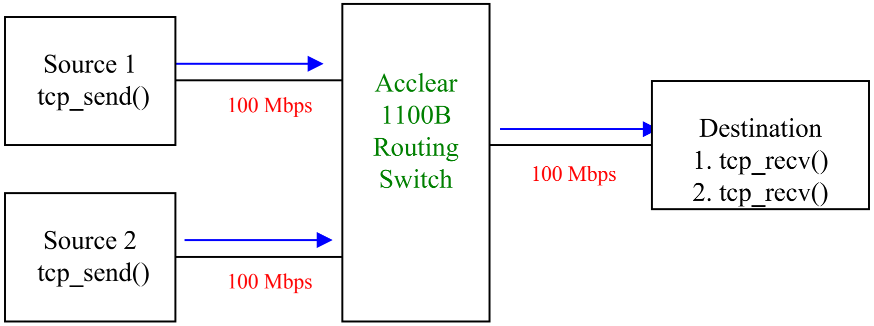
- Implement flow performance enhancement mechanisms **without** introducing software into data forwarding path
  - **Service defined packet processing in a silicon-based forwarding engine**
  - **Dynamic packet classifier**

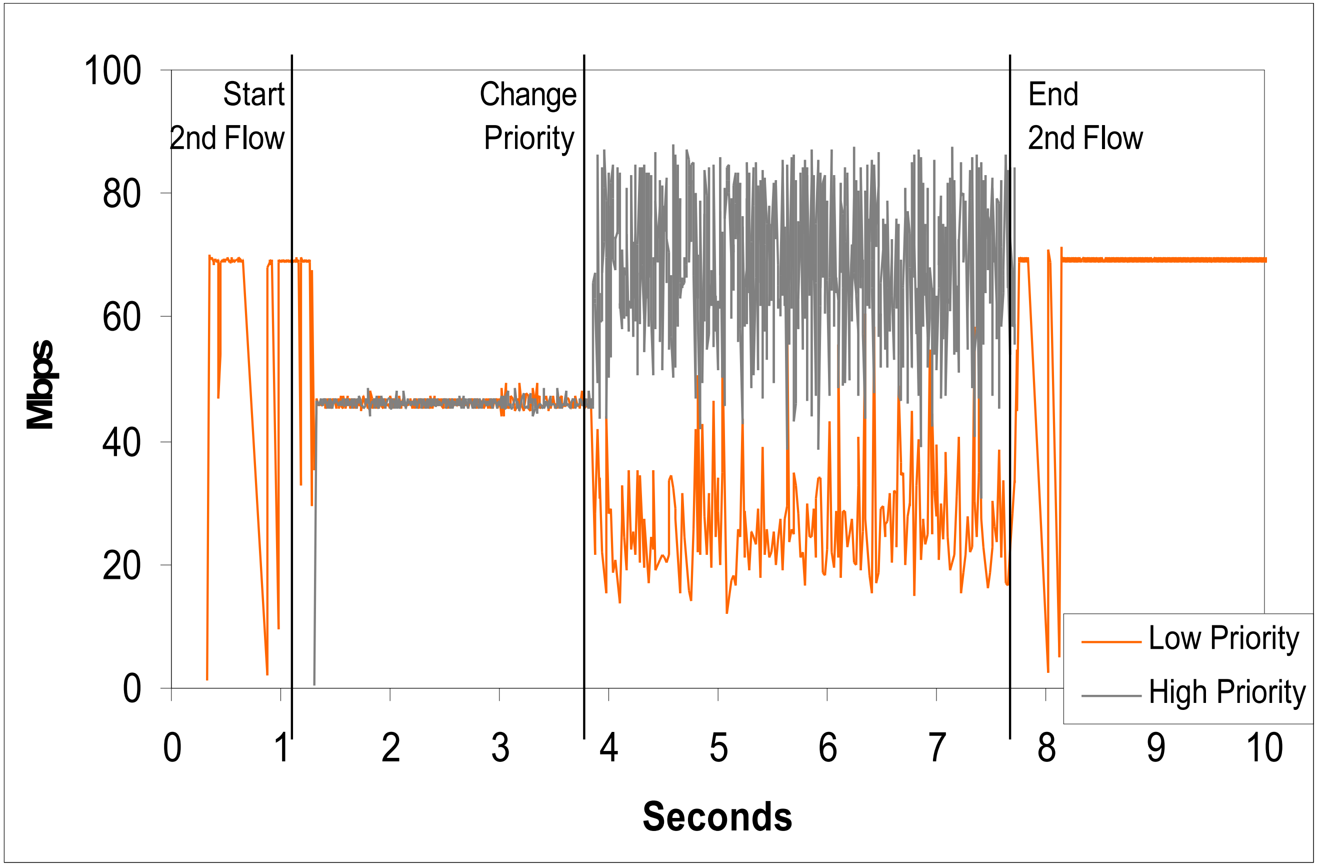
- Rob Jaeger, Jeff Hollingsworth, Bobby Bhattacharjee - University of Maryland

Dynamic - On the Fly Configuration



Experimental Setup




100
80
60
40
20
0
Mbps
0
1
2
3
4
5
6
7
8
9
10
Seconds
Start
2nd Flow
Change
Priority
End
2nd Flow
Low Priority
High Priority

Dynamic Classification

- Identify real-time flows (e.g. packet signature/flowId)
  1. Use CarbonCopy filters to deliver multimedia control protocols to control plane
     - e.g. SIP, H.323. RTCP
     - Determine dynamically assigned ports from control msgs
  2. Use CarbonCopy filters to sample a number of packets from the physical port and identify RTP packets/signature
- Set a packet processing filter for packet signature to:
  - adjust DS-byte OR
  - adjust priority queue

5-tuple Filtering List

- Source Address
- Source Port
- Destination Address
- Destination Port
- Protocol

JFWD 5-tuple Filtering

- Copy the packet to the control plane
- Don't forward the packet
- Set TOS field
- Set VLAN priority
- Adjust priority queue

Dynamic Classification

- **Without** introducing software into data path we performed **Dynamic Classification of flows in a Silicon-Based Gigabit Routing Switch**
  - Introduced a new service to a Gigabit Routing Switch
  - Identified real-time flows
  - Performed policy-based flow behavior classification
  - Adjusted DS-byte value
  - Showed that flow performance can be improved

Nortel’s Openet.lab

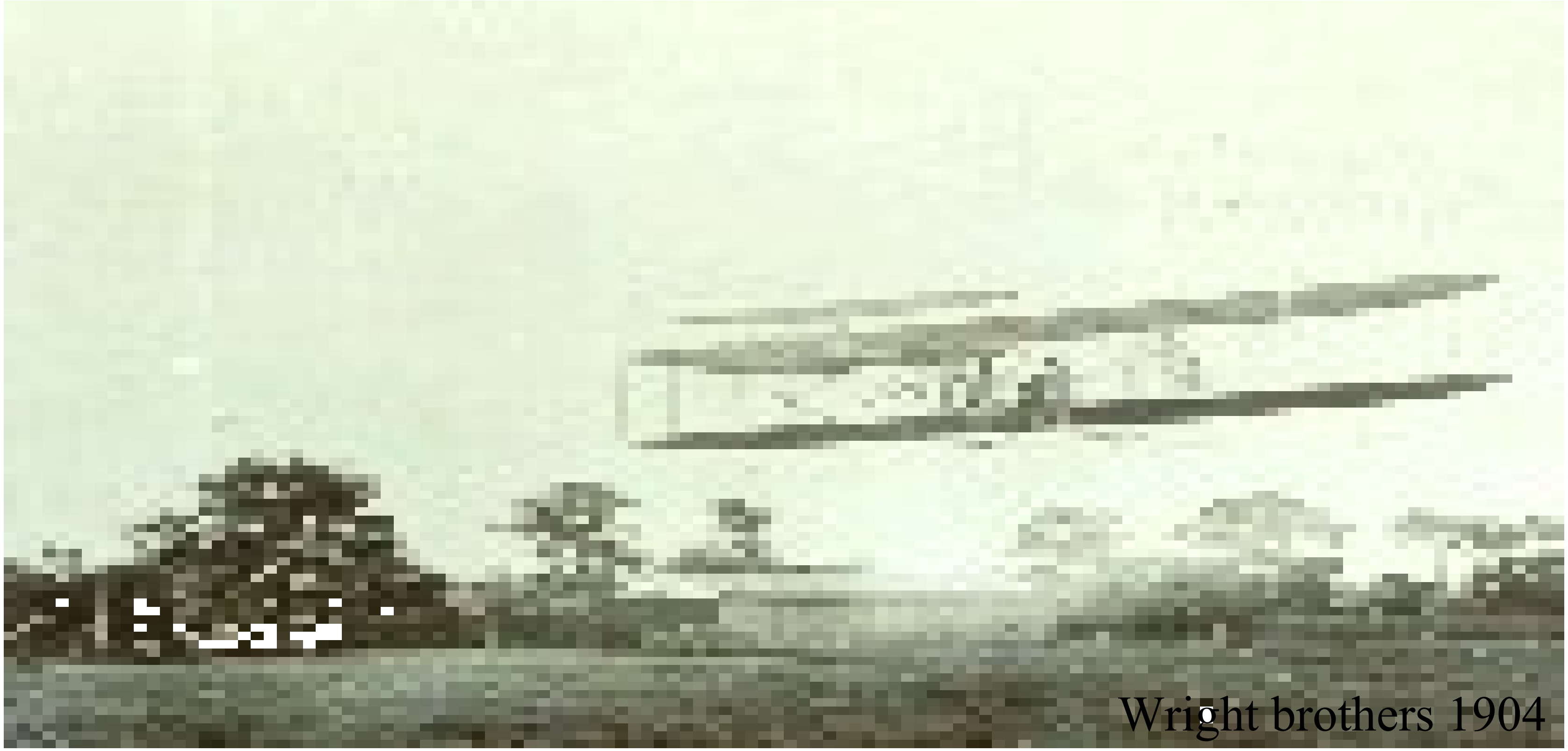
It’s an incubator for service-enabled network nodes and sample services

It provides:

- JVM-emancipated prototypes of Nortel routers
- Java APIs to MIBs
- Java APIs to Forwarding Planes, packet capturing
- A runtime environment for downloaded code

Free downloads from http://www.openetlab.org

Closing remark



Wright brothers 1904

Back then, thrust wasn't a problem; control was.

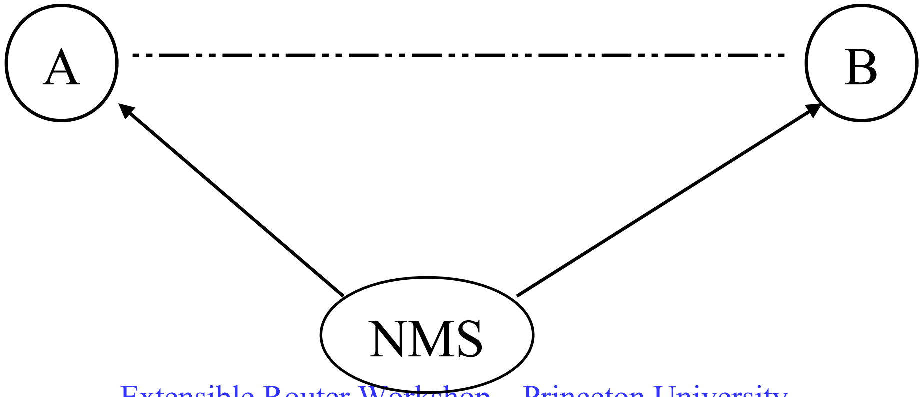
Likewise, network bandwidth isn't the problem, control is. It demands our collective efforts

Q&A

Appendix

Multiple points of view

- It is possible for node A to lose network “visibility” to node B, even though the NMS has visibility to both
- The NMS is the traditional PoV for observing the network
- Being able to move the management PoV out of the NMS and into the managed nodes would help

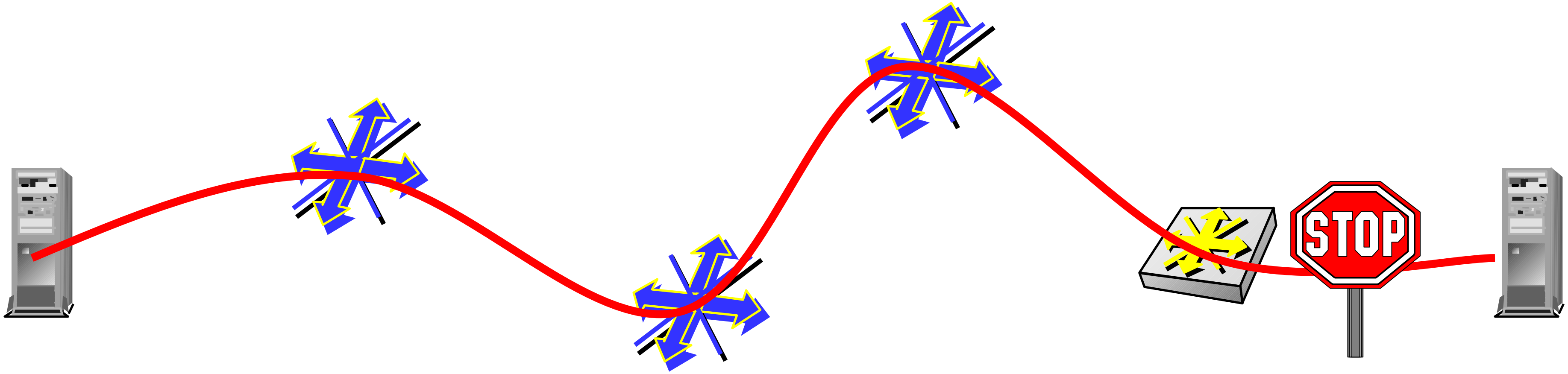


Mobile diagnostics

- Similar to multiple points of view
- Blocking DoS at ingress into the network is best
- Inject mobile agent into the network at the node where the DoS is first detected
- The agent moves from node to node towards the DoS traffic source
- A bit like an immune system ☺

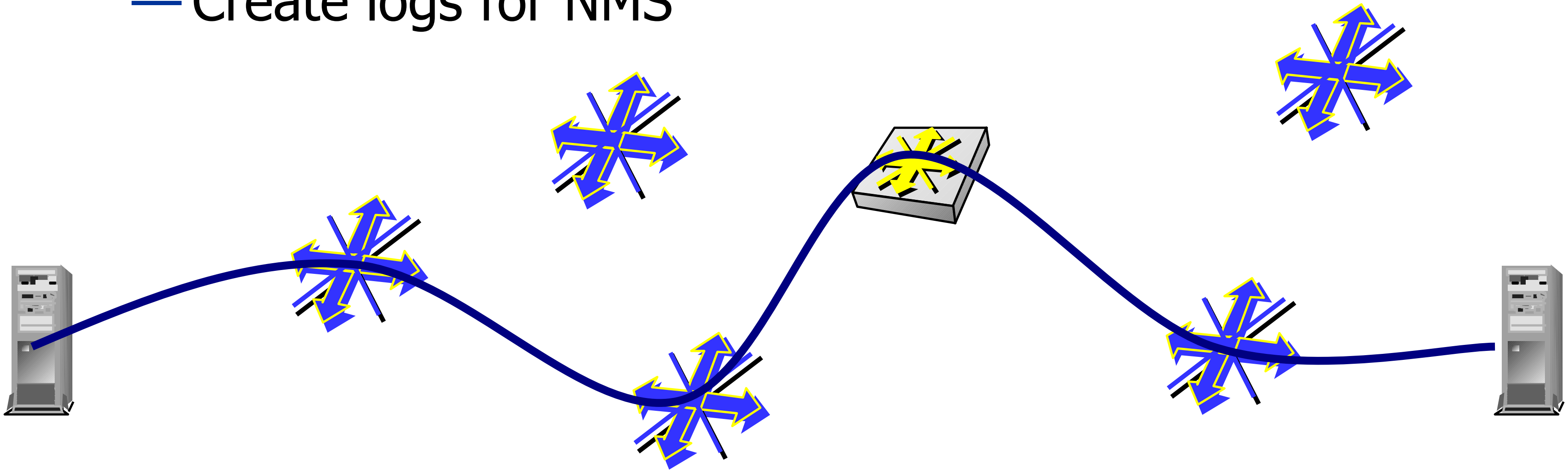
Active Intrusion Detection

- Intruder is identified by Intrusion Detection software
- Intruder signature is identified
- Mobile agent is dispatched in direction of intruder (based on physical port of entry)
- Mobile agent "chases" and terminates intruder (shuts down link, reboot host, notify NMS)



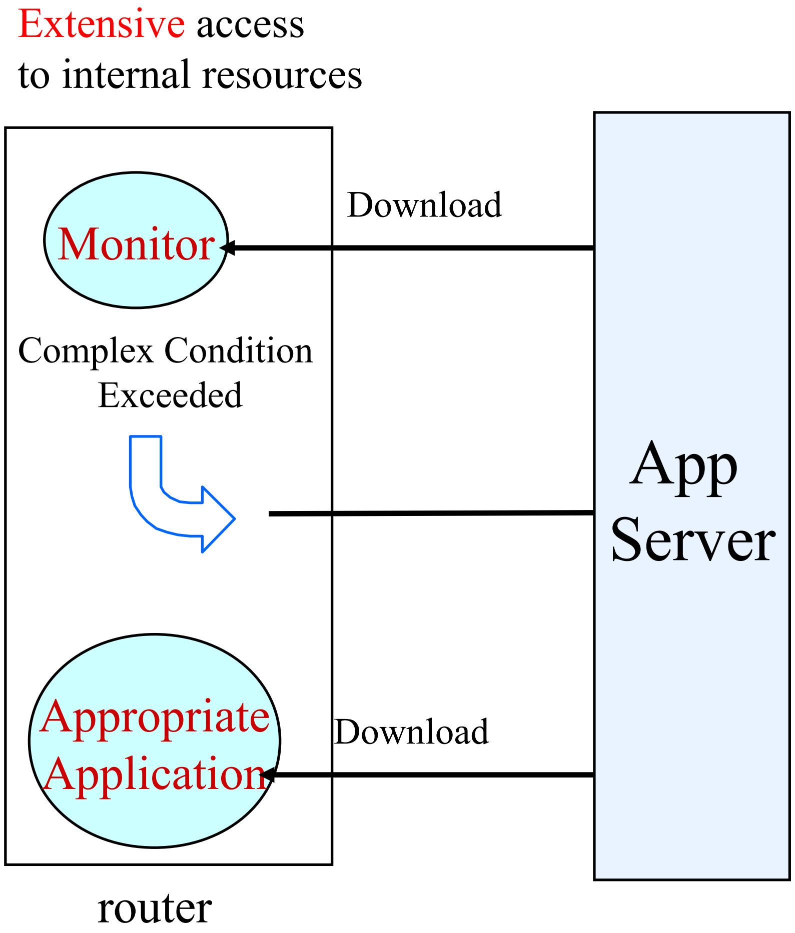
Diagnostic Mobile Agents

- **Automatic trace-route from edge router where problem exists**
  - Each node reached generates a report to NMS
  - Trace-route code “moves” to next node in path
  - Mobile agents identify router health
  - Create logs for NMS



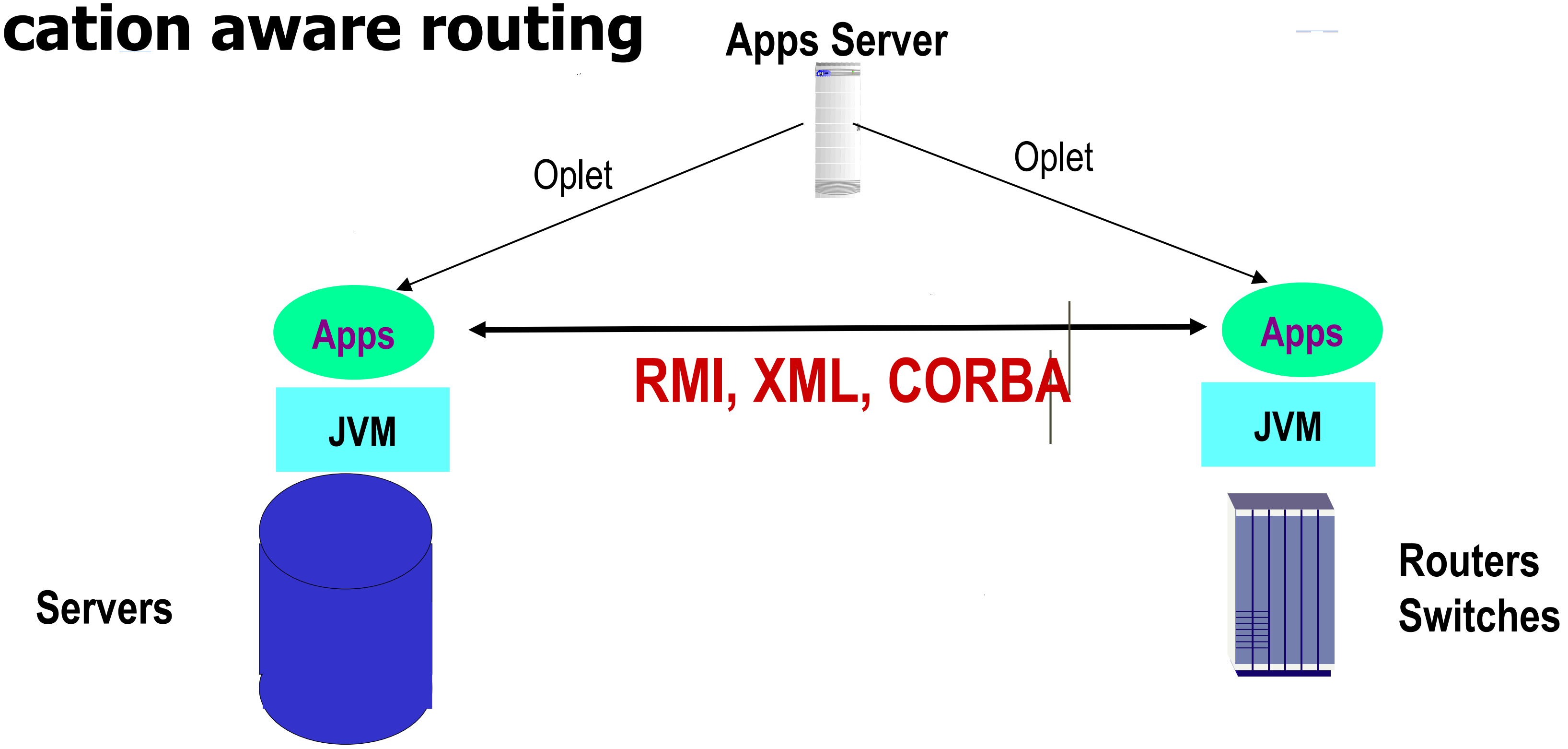
Apps - Routing Relationship

- **Download Oplet Service to the router.**
- **Monitor router locally**
- **Report "events" to App server**
- **Allow Service to take action**
- **Download application**
- **Adjust parameters based on direction from app server**



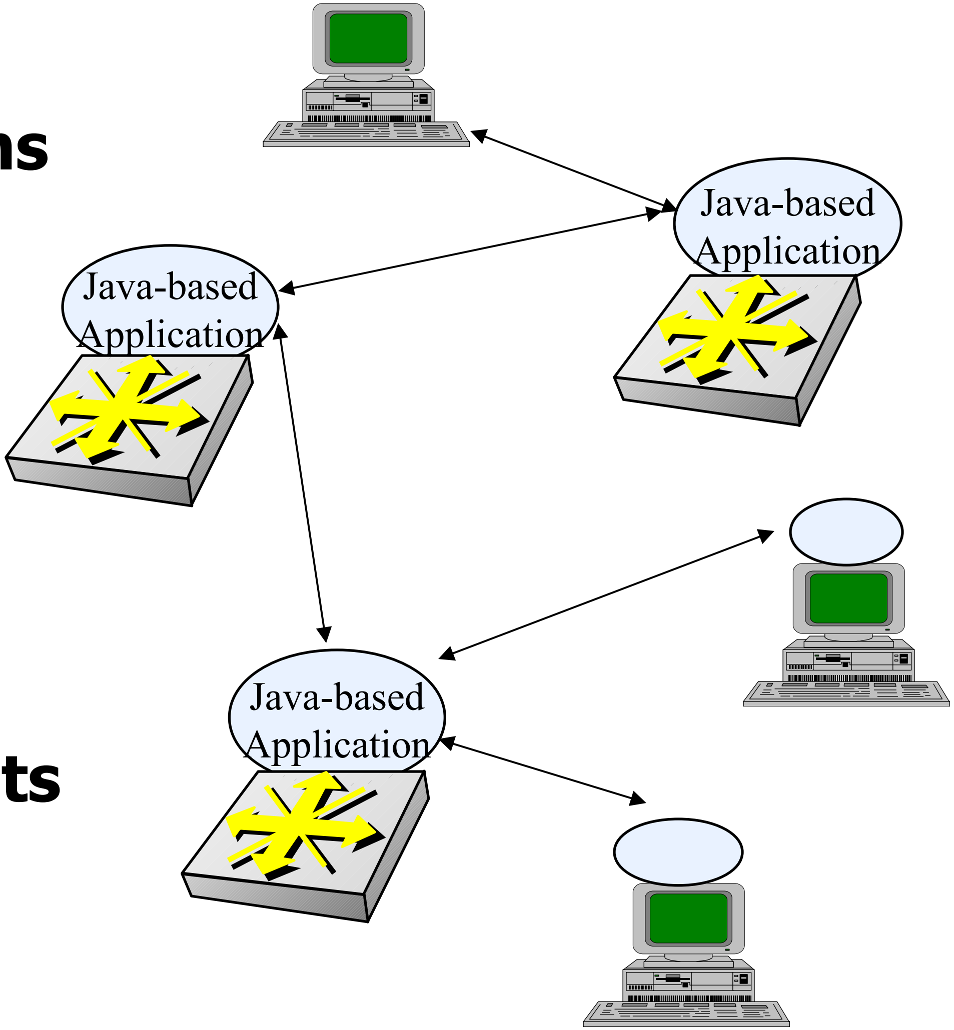
Collaboration with Applications

- New paradigm of distributed applications
- Network devices collaborating with applications
- Application aware routing



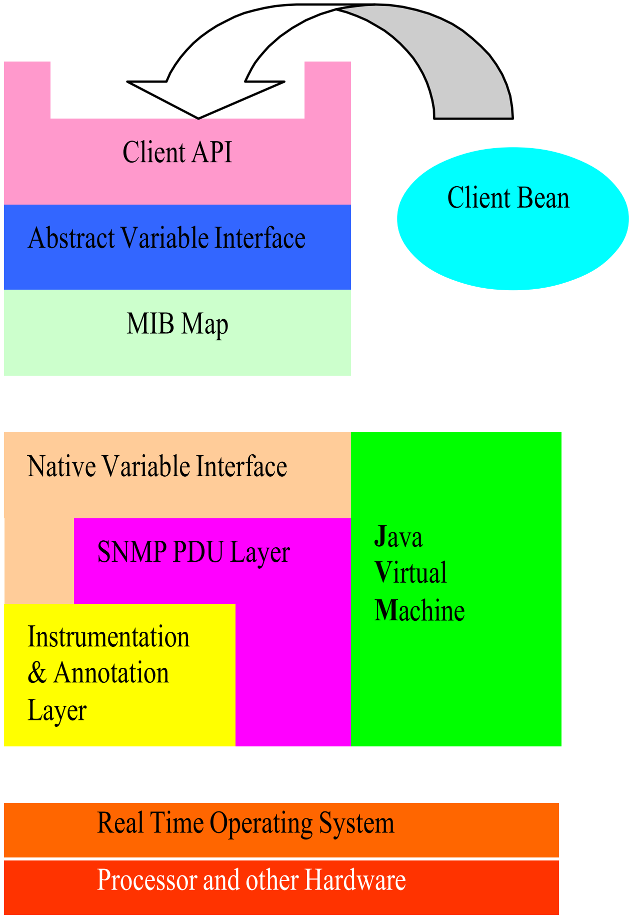
Router Server Collaboration

- **Supports distributed computing applications in which network devices participate**
  - router to router
  - server to router
- **Supports Intelligent Agents**
- **Supports Mobile Agents**



MIB API Example

- API uses a MIB Map to dispatch requests to variable access routines
- Different parts of the MIB tree can be serviced by different mechanisms
- Two main schemes:
  - An ad hoc interface to the SNMP instrumentation layer
  - A generic SNMP loopback


Client API
Abstract Variable Interface
MIB Map
Client Bean
Native Variable Interface
SNMP PDU Layer
Java
Virtual
Machine
Instrumentation
& Annotation
Layer
Real Time Operating System
Processor and other Hardware

Strong Security in the New Model

- **The new concept is secure to add 3rd party code to network devices**
  - Digital Signature
  - Administrative "Certified Optlet"
  - No access out of the JVM space
  - No pointers that can do harm
  - Access only to the published API
  - Verifier - only correct code can be loaded
  - Class loader access list
  - JVM has run time bounds, type, and execution checking