GENI Backbone Working Group
NSF and GPO (June 26, 2007)

Dan Blumenthal, Dean Casey, Patrick Crowley, T.V. Lakshman, Bryan Lyles, Sarit Mukherjee, Nick McKeown, Jen Rexford, Jon Turner, and Hui Zhang

Agenda

- Introduction
  - High-level backbone architecture
  - Related GENI Design Documents
- Packet processing system
  - Reference design and low-level software
  - Prototype MetaRouter system
- Management software
  - Component manager, gateways, libraries
  - Prototypes of VINI and Meta Management
- Open issues and discussion

Some Comments on the Presentations

- Some topics are spread throughout the talks
  - Example experiments
  - Budget estimates from the WBS
  - Circuit processing system (for future meeting)
  - Relationships to the GENI Design Documents
- Topics are presented in a bottom-up style
  - Emphasis on prototype systems
  - And their connection with the architecture
  - To make the discussion more concrete
- Quite a bit of time allocated for discussion
  - To make sure we address your questions

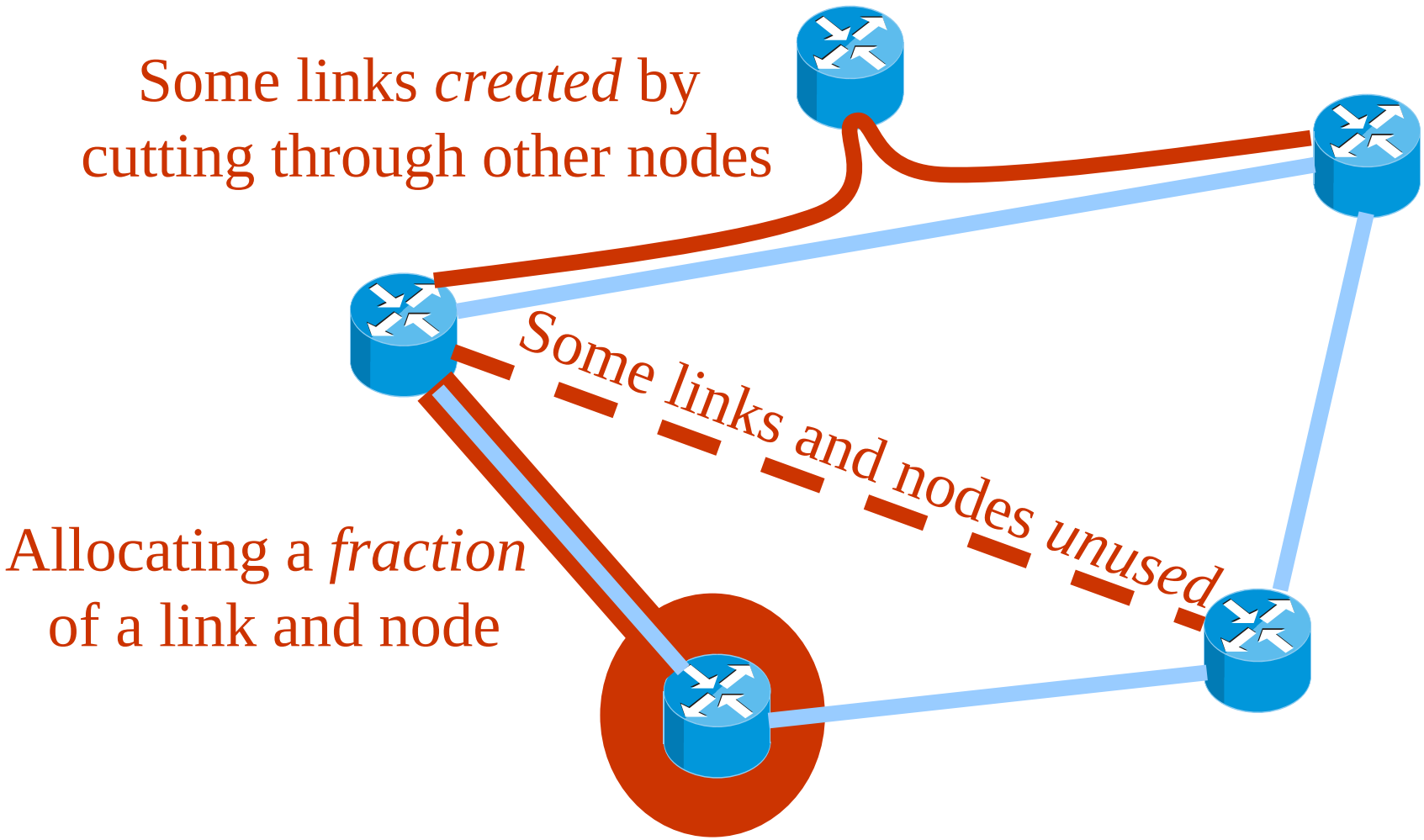
GENI Backbone Requirements

- Programmability
  - Flexible routing, forwarding, addressing, circuit set-up, ...
- Isolation
  - Dedicated bandwidth, circuits, CPU, memory, disk
- Realism
  - User traffic, upstream connections, propagation delays, equipment failure modes
- Control
  - Inject failures, create circuits, exchange routing messages
- Performance
  - High-speed packet forwarding and low delays
- Security
  - Preventing attacks on the Internet, and on GENI itself

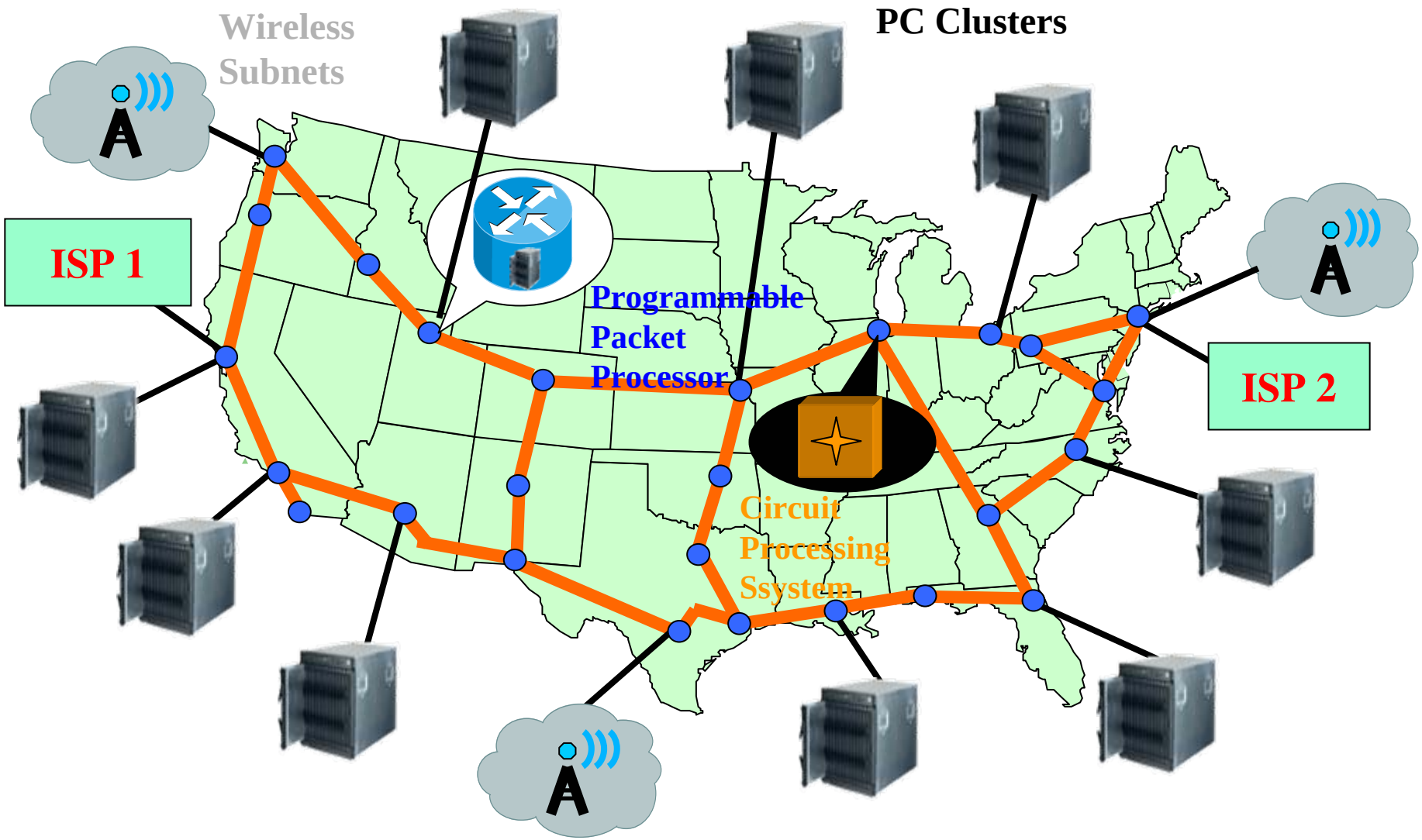
A Researcher's View of GENI Backbone

- Virtual network topology
  - Nodes and links in a particular topology
  - Resources and capabilities per node/link
  - Embedded in the GENI backbone
- Virtual packet processor and virtual circuits
  - To evaluate new architectures (routing, switching, forwarding, addressing, framing, grooming, ...)
- GENI backbone capabilities evolve over time
  - To realize abstraction at finer detail
  - To scale to large number of experiments

Creating a Virtual Topology



GENI Backbone



GENI Backbone Phases

- Phase 0: general purpose processors
  - General purpose processors connected to switch
  - Virtualization platform for multiple virtual routers
- Phase 1: high-end programmable packet processor
  - General-purpose processors, NPs, FPGAs, and line cards
  - Faster packet processing and line-card cut through
- Phase 2: reconfigurable optics
  - Cross-connect and off-the-shelf framer/groomer
  - True circuits and bypass of packet processor
- Phase 3: programmable optics
  - Dynamic optical switch with programmable framer
  - Experimental flexibility for framing, grooming, set-up, ...

Feasibility of the Design

- Industrial trends and standards
  - Advanced Telecom Computing Architecture (ATCA)
  - Network processors and FPGAs
  - SONET cross connects and ROADMs
- Open-source networking software
  - Routing protocols, packet forwarding, network address translation, diverting traffic to an overlay
- Existing infrastructure
  - Experiences from PlanetLab, Orbit, Emulab, ...
  - National Lambda Rail and Internet2 backbones
  - Ongoing work on the MetaRouter and VINI

Budget Estimates from the WBS

- Packet processing hardware (1.2.2.1.1.1)
  - $14M, mostly equipment costs
- Packet processing software (1.2.2.1.1.2)
  - $9M, mostly software development and testing
- Backbone management aggregate (1.2.2.4)
  - $8M, mostly software development and testing
- Internet exchange point deployment (1.2.3)
  - $20M, mostly fees for the bandwidth

Existing GENI Design Documents

- General GENI facility
  - GDD-06-27: "GENI Topology Design"
    - *Why ~25 nodes, why ~200 edge sites*
  - GDD-06-47: “Life of a Packet within a GENI Experiment”
    - *Connecting sites, multiplexing headers, cut-throughs*
- Backbone node architecture
  - GDD-06-09:“A Proposed Architecture for the GENI Backbone Platform”
    - *Packet processing system*
  - GDD-06-26: “GENI Backbone Network Node Architecture: Requirements and Architecture”
    - *Circuit processing system*

Existing GENI Design Documents (Cont.)

- Backbone management software
  - GDD-06-25: “Backbone Software Architecture”
    - *Initial outline of the backbone software (obsoleted)*
  - GDD-06-31: "In VINI Veritas: Realistic and Controlled Network Experimentation”
    - *Design, implementation, & evaluation of VINI prototype*
  - GDD-06-36: “GENI Backbone Run-Time Software for Experimenters”
    - *Experiment libraries, gateways, Internet connections*
  - GDD-06-37: “Meta-Management System for GENI”
    - *Boot-strapping of communication for management*

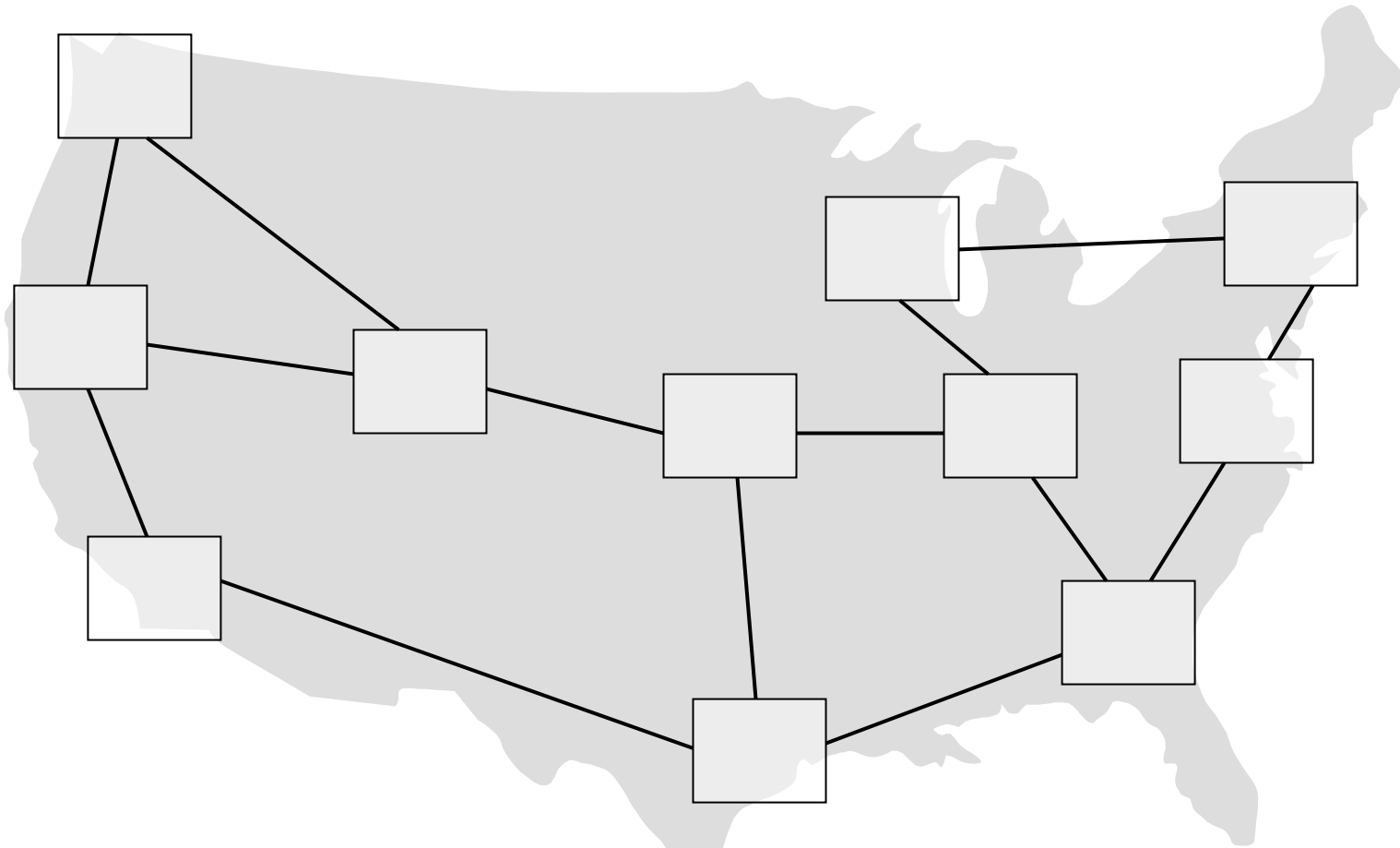
GENI Backbone Software

Goals of This Presentation

- Architectural components
  - Component manager
  - Management aggregate
- Experiment support
  - Installing forwarding-table entries in data plane
  - Sending data packets to/from the Internet
  - Sharing BGP sessions with neighboring ISPs
- Prototype systems
  - VINI overview and status
  - Meta-management system
- Example experiments

VINI Physical Deployment

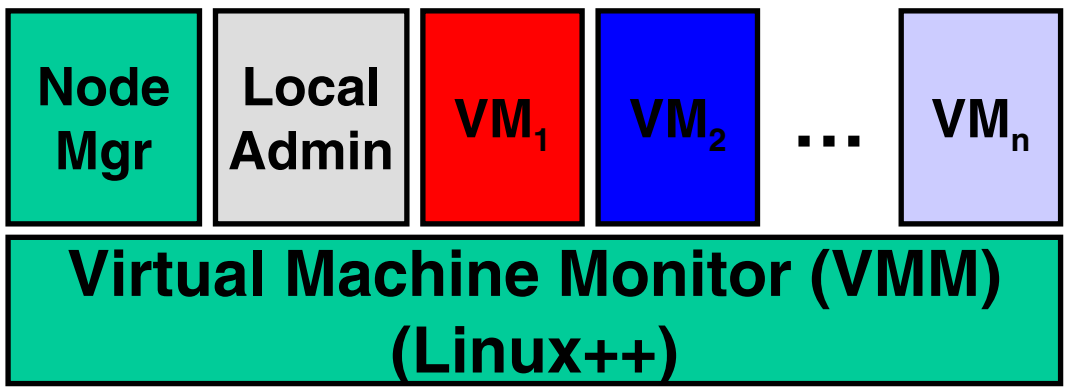
http://www.vini-veritas.net



Deployed high-end servers in National Lambda Rail and Abilene, and PoPs in Seattle and Virginia

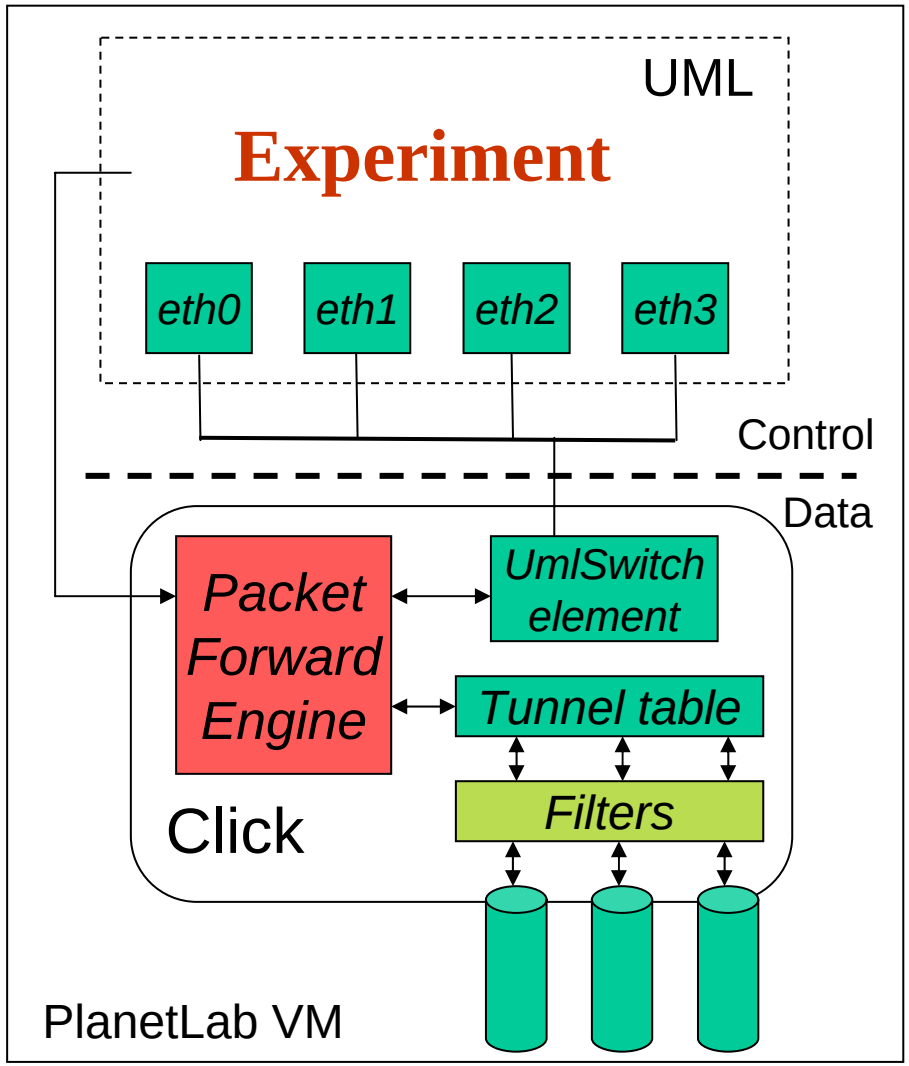
VINI Node Software: Underlying Substrate

- Started with the PlanetLab software
  - Virtual machines with name-space isolation
  - Each has “root” in its own VM, and can customize
  - Reserve CPU and bandwidth per experiment



VINI Node Software: Deployed Base

- Added net virtualization
  - Virtual interfaces
  - Bound to tunnels
- Initially using User-Mode Linux (UML)
  - Click packet forwarding
  - In user space (200 Mbps)
- Installing the FIB
  - Click Forwarding Element Abstraction (FEA)
  - Linux iptables
- In deployment and use



VINI Node Software: Faster Forwarding

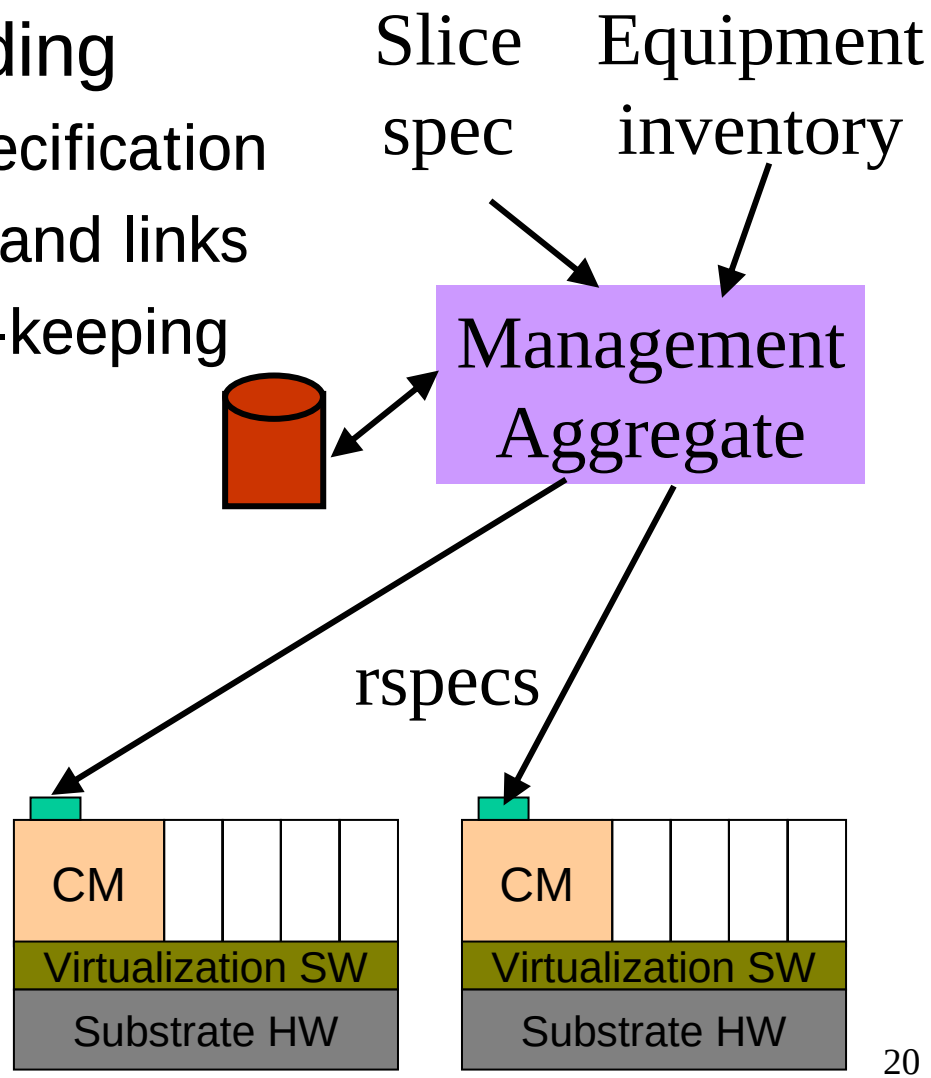
- Virtualized network stack in Linux
  - Network views that are bound to processes
  - Separate kernel forwarding tables per view
  - Supports virtualization *within* an experiment
- Status of enhancements to Linux
  - Current doing testing and performance studies
  - Scheduled for deployment at summer's end
  - Working to incorporate changes in mainline Linux
- Hardware support through FPGAs and NPs
  - Nick McKeown's NetFPGA project
  - Jon Turner's MetaRouter project

GENI Node Software: Experiment Libraries

- Wide range of researchers
  - Using the backbone just for the connectivity
    - No backbone processing of the data packets
  - Experimenting only in the control plane
    - Running conventional IPv4/IPv6/Ethernet data plane
  - Building their own data plane
    - In Click, in C, in Verilog/VHDL
- Multiple ways to install forwarding tables
  - Raw access to the network processor
  - Shadowing the Linux iptables
  - Click Forwarding Element Abstraction (FEA)
  - IETF ForCES standard

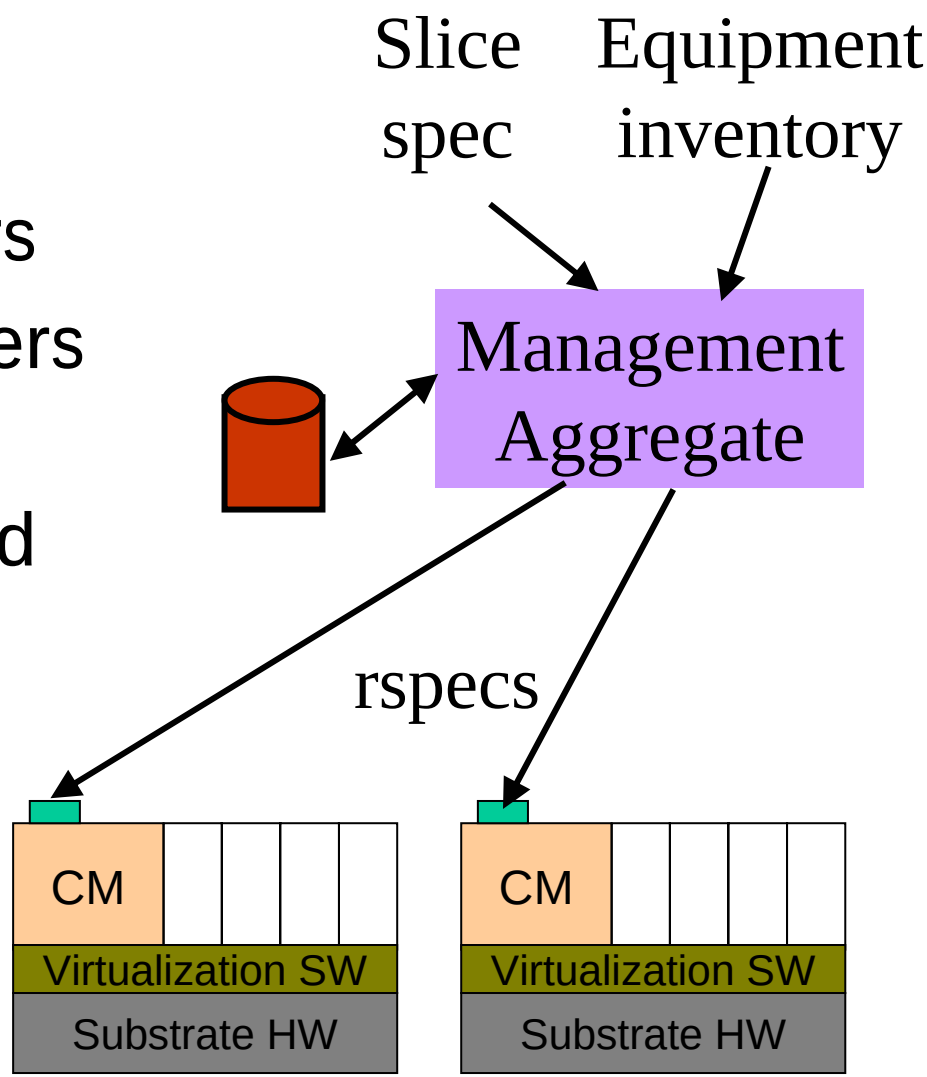
VINI Management Software

- Management aggregate
  - Virtual network embedding
    - Given slice backbone specification
    - Selects substrate nodes and links
    - Admission control, book-keeping
  - Slice creation
    - Generates and “rspec” for each component
    - Sends to component
- Current status
  - Simple Ruby scripts
  - Working on DB



VINI Management Software

- Component manager
  - Creates the virtual machine and containers
  - Associates the containers with tunnels
  - Configures the CPU and bandwidth schedulers
- Current status
  - Simple scripts



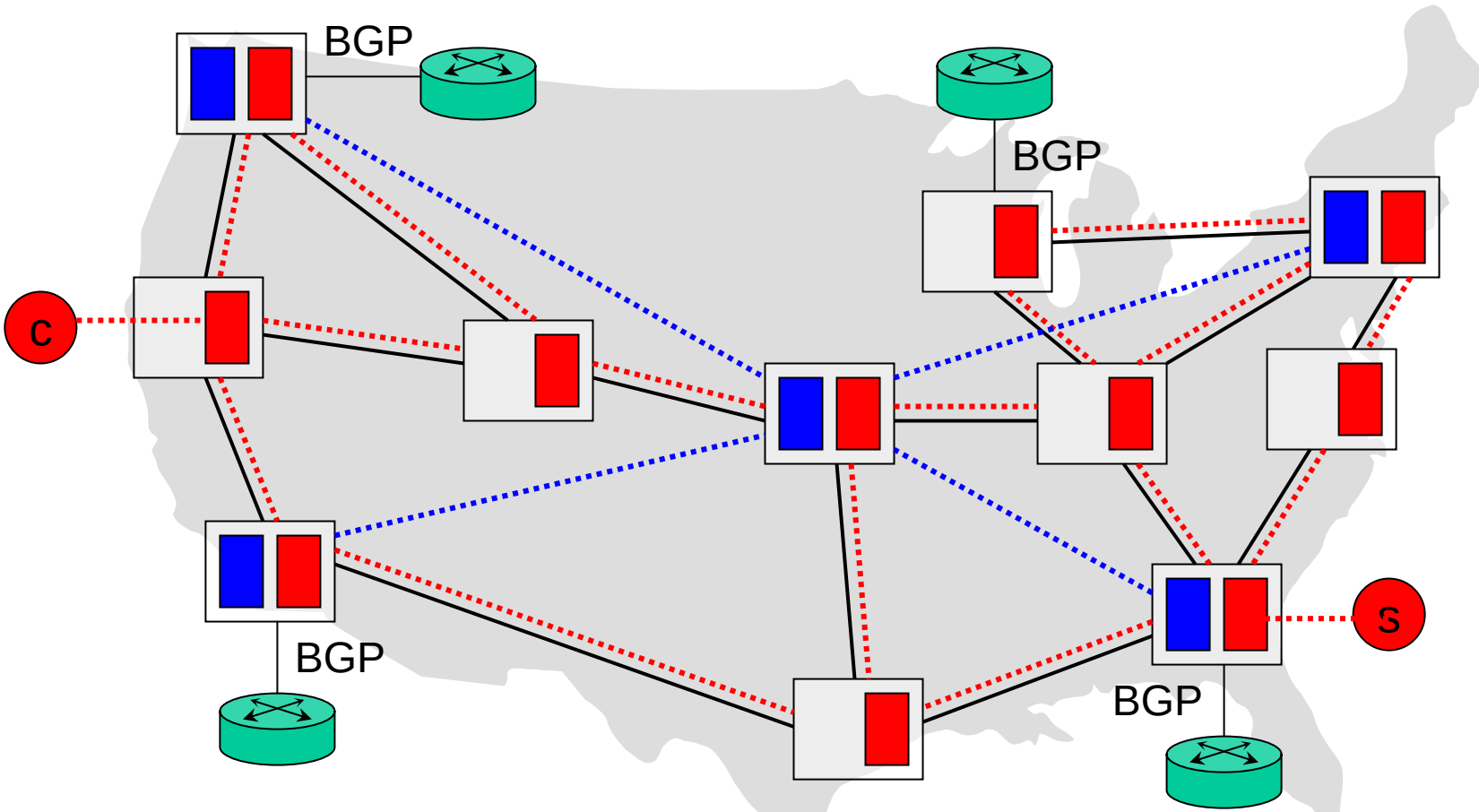
Toward the GENI Management Software

- VINI has a simple initial rspec design
  - Node has CPU resources, link has bandwidth
  - Best-effort or guaranteed fixed resources
  - No QoS support from intermediate switches
- Packet processor has more sophisticated rpsec
  - Multiple types of cards in the MetaRouter
  - Shared vs. dedicated network processors
  - Additional resources like SRAM and TCAM space, inter-chassis bandwidth, switch-fabric resources
- Component manager monitoring
  - Reporting on node and sliver status

Connecting to the Legacy Internet

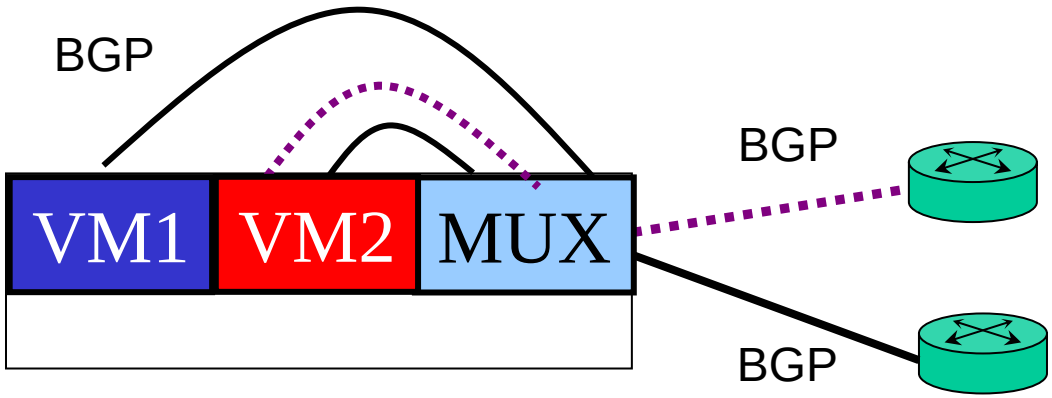
- Experiments need to connect to the Internet
  - Allow users from non-GENI sites to “opt in”
  - Access content and applications on the Internet
- VINI connecting to end users
  - OpenVPN server running in an experiment
  - Application-level proxy running in an experiment
- VINI connecting to the rest of the Internet
  - Network address translation at boundary point
  - To share a limited public address space (/20)
  - And to ensure return traffic reaches experiment

External Routing Adjacencies



Experiments can participate in Internet routing (to control outbound traffic, or announce prefixes)

BGP Mux: Sharing a BGP Session



- Provides illusion of dedicated BGP sessions
  - While maintaining a single session per neighbor
- Filters route updates to protect Internet
  - Only address blocks the experiment “owns”
  - Within some limits on update frequency
- Initial prototype as an extension to Quagga

A General Need for Gateways

- Some resources are easy to share
  - CPU: time sharing
  - Bandwidth: time sharing, frequency sharing
- But logical resources are harder to share
  - BGP session to a neighboring ISP
  - Command-line access to optical equipment
- Gateways to arbitrate access
  - Provide the illusion of dedicated access
  - While providing resource isolation
  - And protecting against misbehavior, crashes, ...

Getting VINI Closer to the Optics

- VINI today is essentially an overlay
  - VINI nodes are connected by IP
  - Typically over a single IP-layer hop
  - Though the substrate reroutes us around failures
- Going forward: ongoing plans with Abilene
  - Connect VINI nodes directly to Ciena CoreDirector
  - Establish dedicated circuits between VINI nodes
- Planning to follow a phased approach
  - Initially, a fixed substrate topology
  - Later, allow per-experiment topologies
  - And perhaps later to span ESnet and GEANT

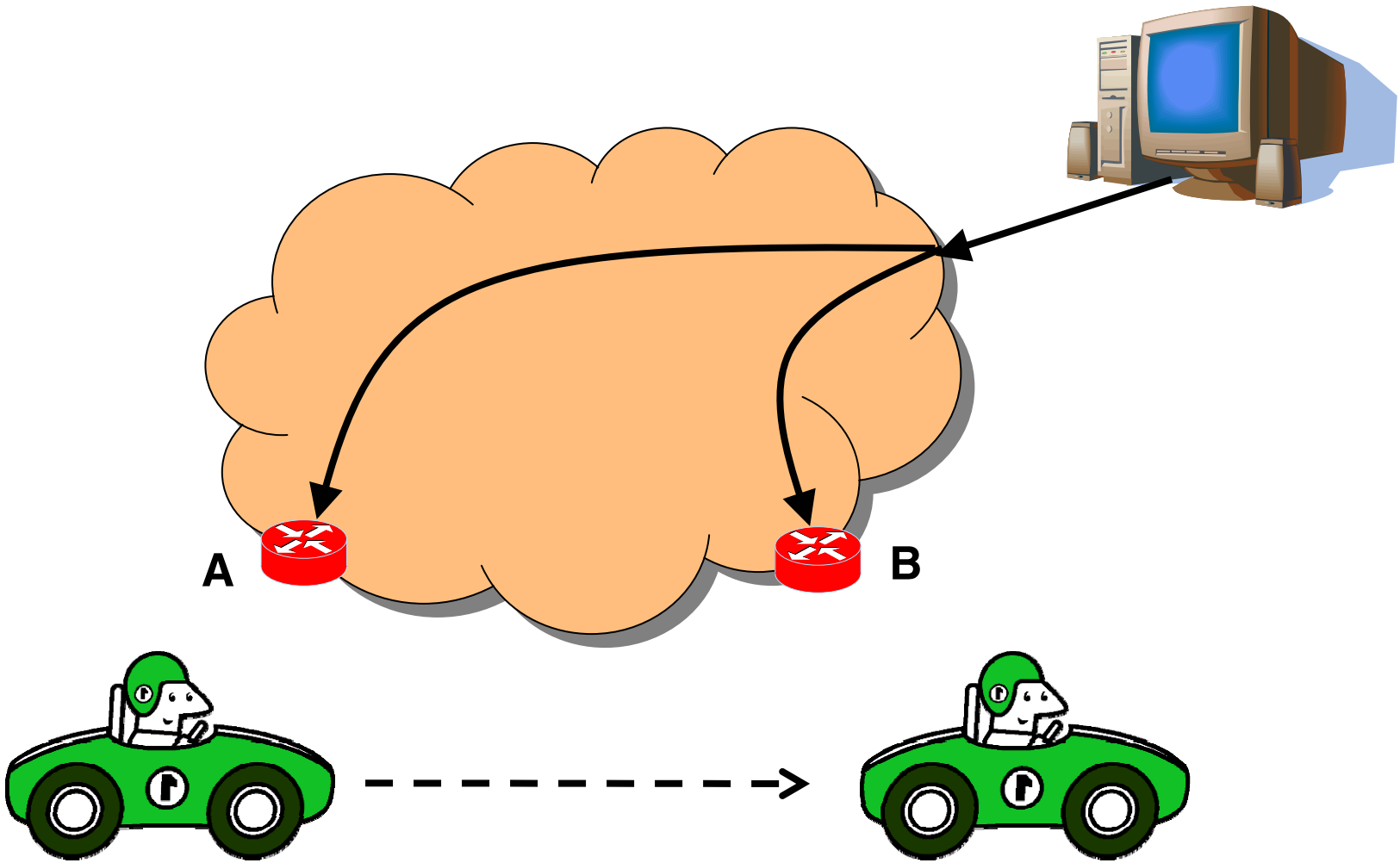
Experiments Run on VINI:
Example of Joint ORBIT/VINI Experiment

Example VINI Experiments

- Evaluating routing-protocol convergence
  - Web download during link failure in XORP, Quagga
- Network-layer support for overlay services
  - Overlay forwarding and notification in data plane
- Piggybacking diagnostic data on packets
  - Data plane support for network troubleshooting
- Scaling Ethernet to a large enterprise
  - Flat addressing and hash-based location resolution
- **Backbone support for mobile hosts**
  - **Injecting end-host addresses in to OSPF**

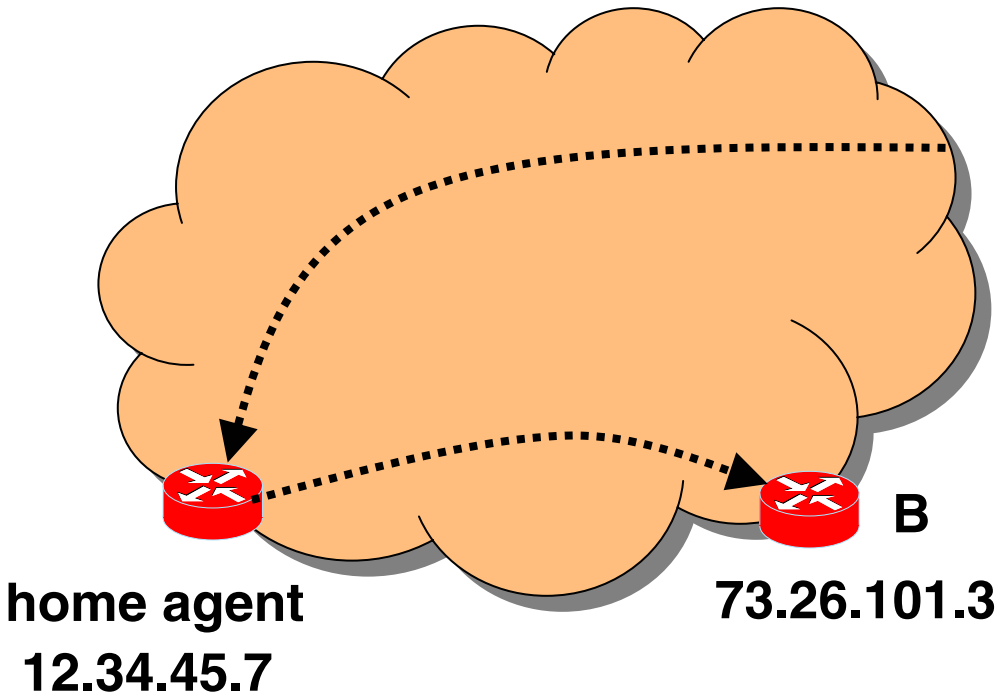
Mobility Challenges

- Seamless transmission to a mobile host



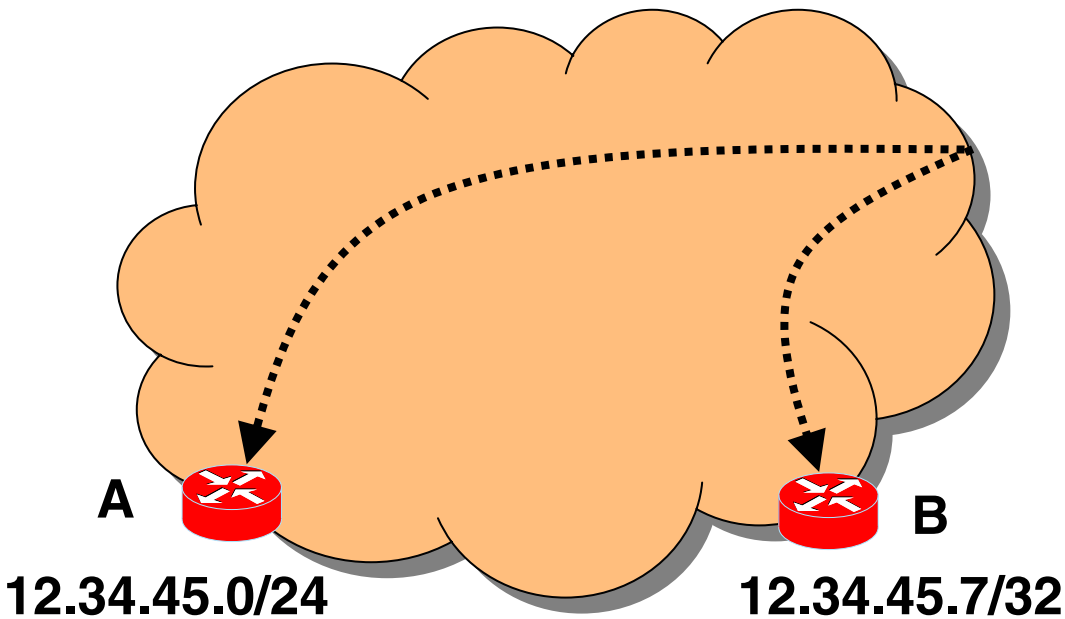
No Backbone Changes: Mobile IP

- Mobile node has two addresses
  - Home address: fixed
  - Care-of address: changes as the host moves
- Packets relayed through the home agent



No Changes to Hosts: Route Injection

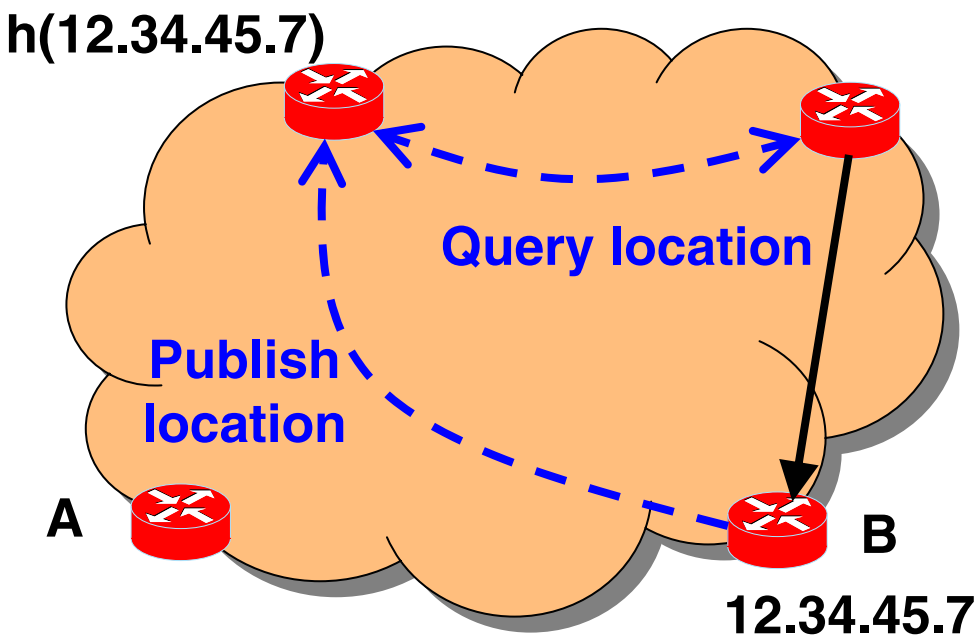
- Mobile node has a single, persistent address
- Address injected into routing protocol (e.g., OSPF)
- But, flat addressing causes scalability challenges



Similar to approach used in the Boeing Connexion service…

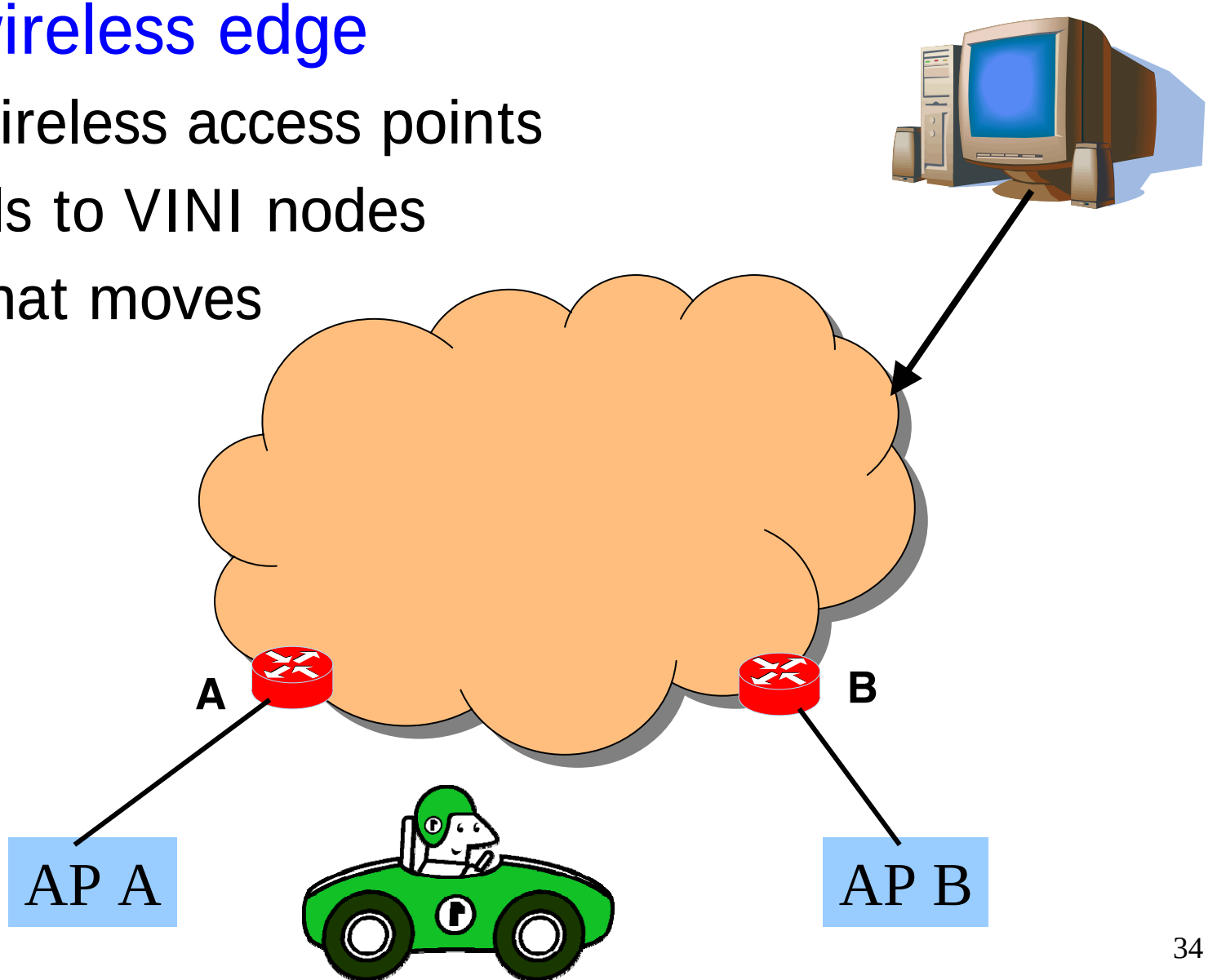
Scalable Support for Flat Addressing

- Store location info at a small set of nodes
  - Fetch based on hash of address
- Cache location info at ingress node
  - Cut-through to send traffic directly to mobile node



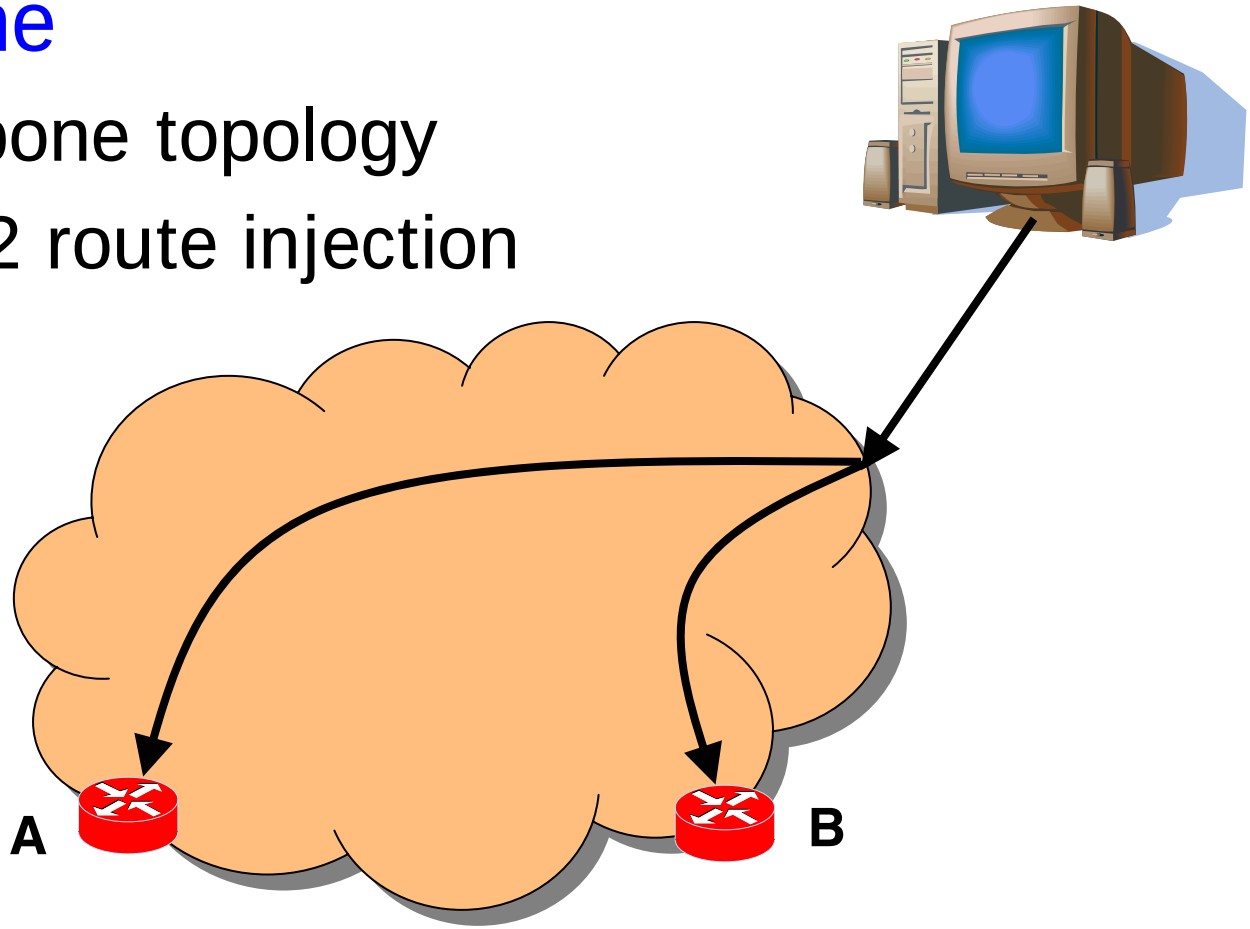
Joint Orbit and VINI Experiment

- Orbit: wireless edge
  - Two wireless access points
  - Tunnels to VINI nodes
  - Host that moves



Joint Orbit and VINI Experiment

- VINI: backbone
  - Abilene backbone topology
  - OSPF with /32 route injection



Joint ORBIT and VINI Experiment

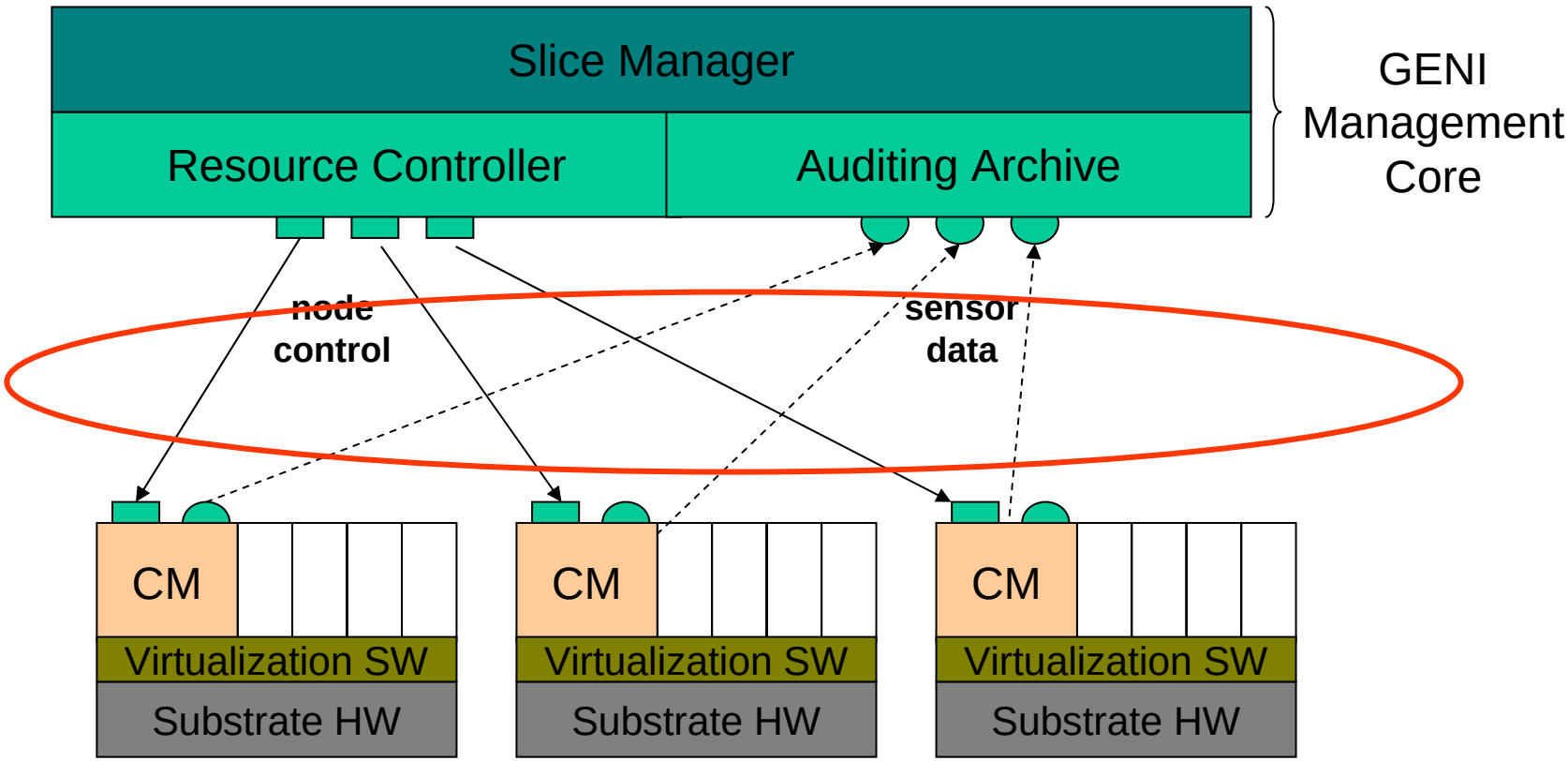
- Experiment/demo
  - Download a video stream from a server
  - While the wireless node is moving
  - Observe the quality of the video stream
- Current status
  - Joint experiment with route injection
  - Separate testing of hash-based scheme
  - Plan to experiment with the hash-based scheme
- Lessons learned
  - Support for non-IP protocols
  - Connecting to nodes behind firewall/NAT

Meta Management System

Zheng Cai, Hemant Gogineni,
David A. Maltz, T. S. Eugene Ng,
Hong Yan, and Hui Zhang

How Does GMC Communicate with CMs?

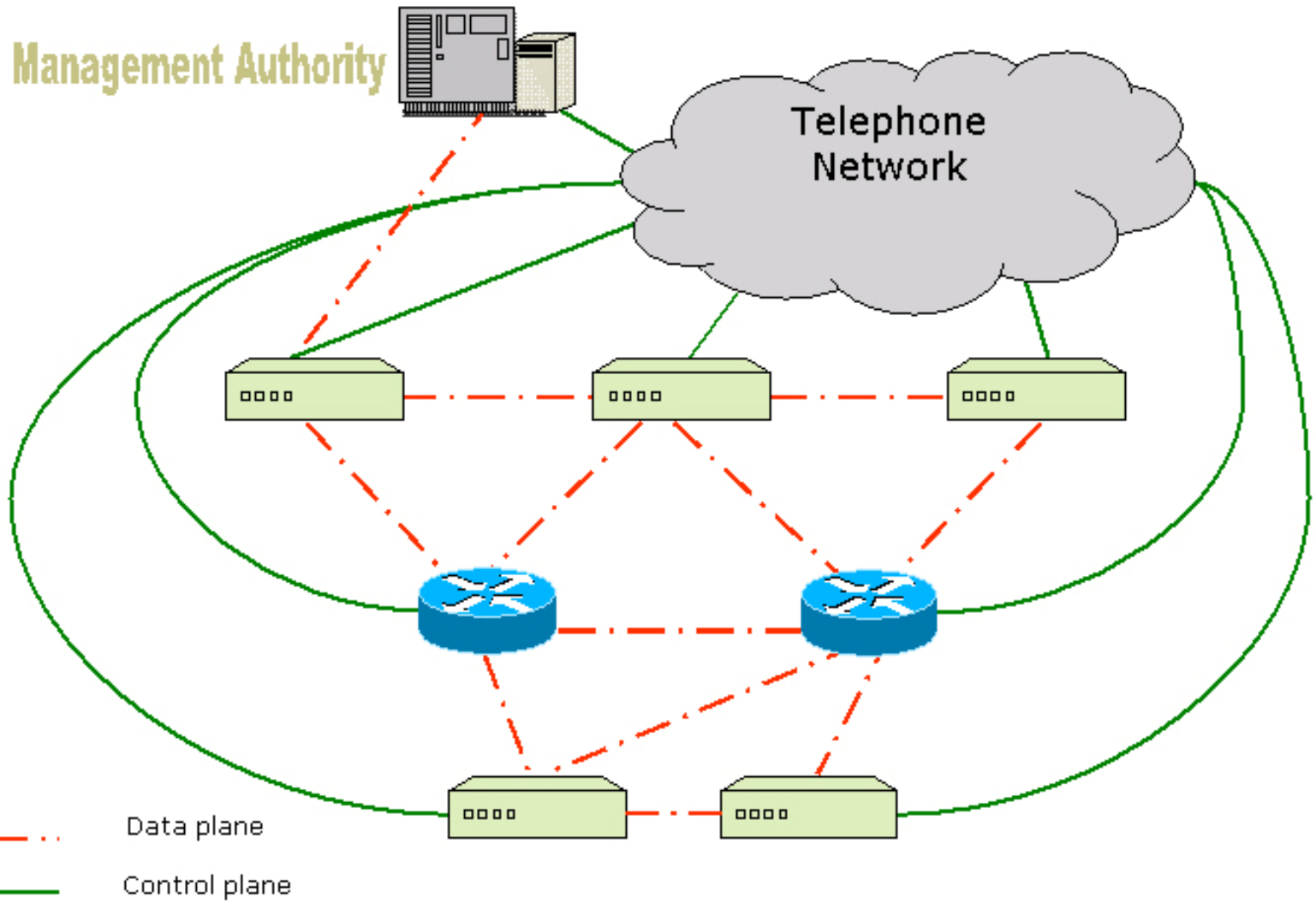
- Fundamental architectural issue that affects overall network manageability and security



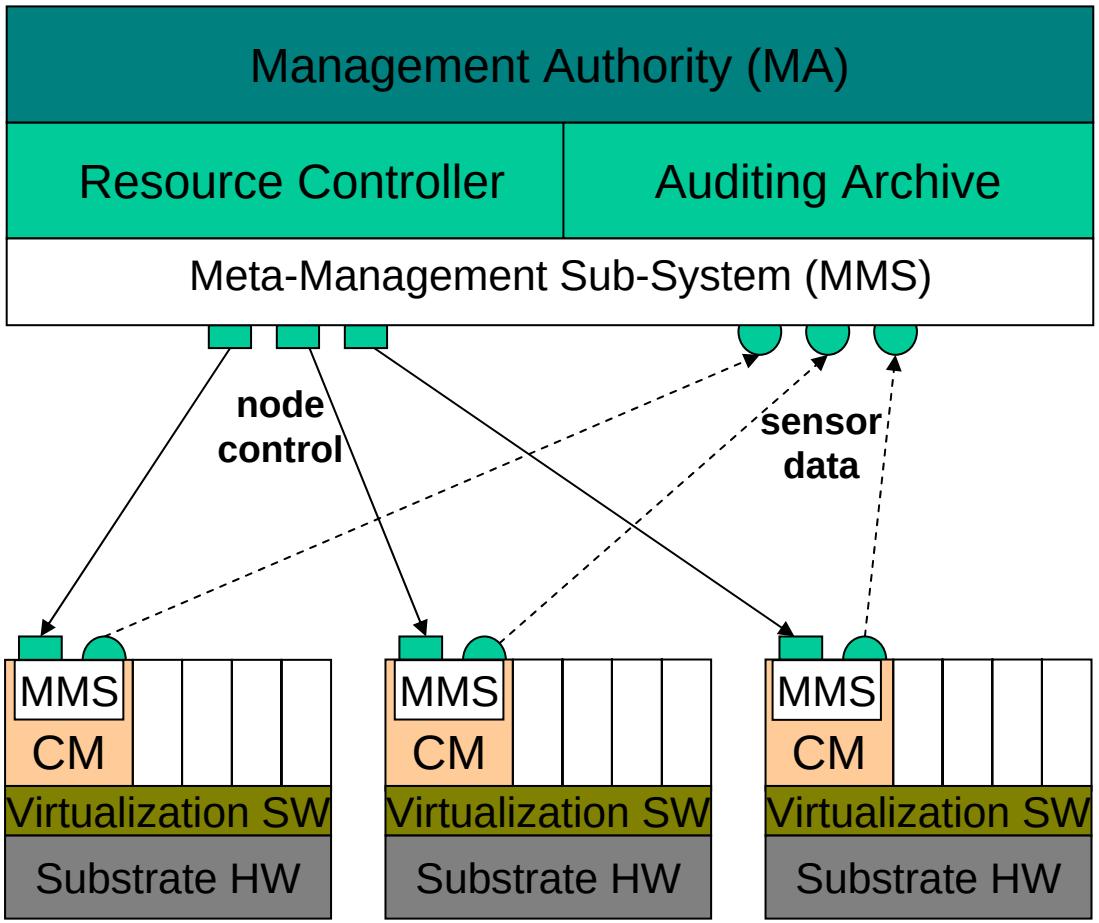
Example GMC Management Scenarios

- Remote router failure diagnosis and reboot
  - Remote diagnosis of a packet processor
  - If necessary, GMC may reboot the node
- Network wide failure diagnosis and reboot
  - Troubleshooting during service disruption
  - Rebooting the entire network, if necessary
- Network maintenance
  - Temporarily remove a node for maintenance
- Network service migration
  - Migrate native in-band service from one (e.g. IPv4) to another (e.g. IPv6)

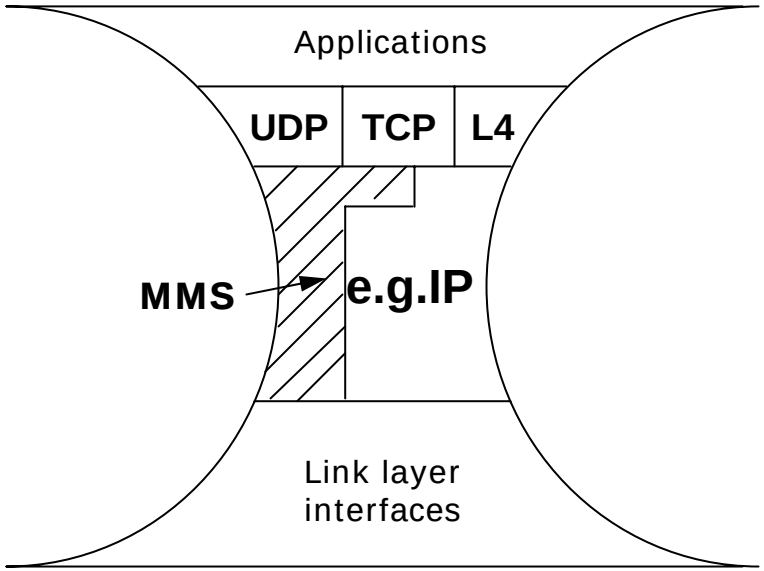
Today: Out-of-Band Management



GENI Meta-Management Sub-System



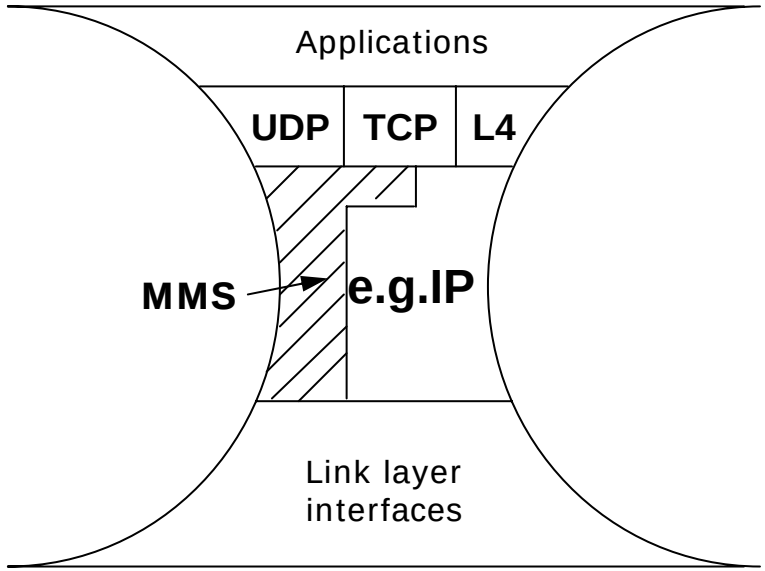
A Meta-Management System (MMS)



- Secure and robust management channel
  - As long as there is physical network connectivity
  - Does not need a separate network

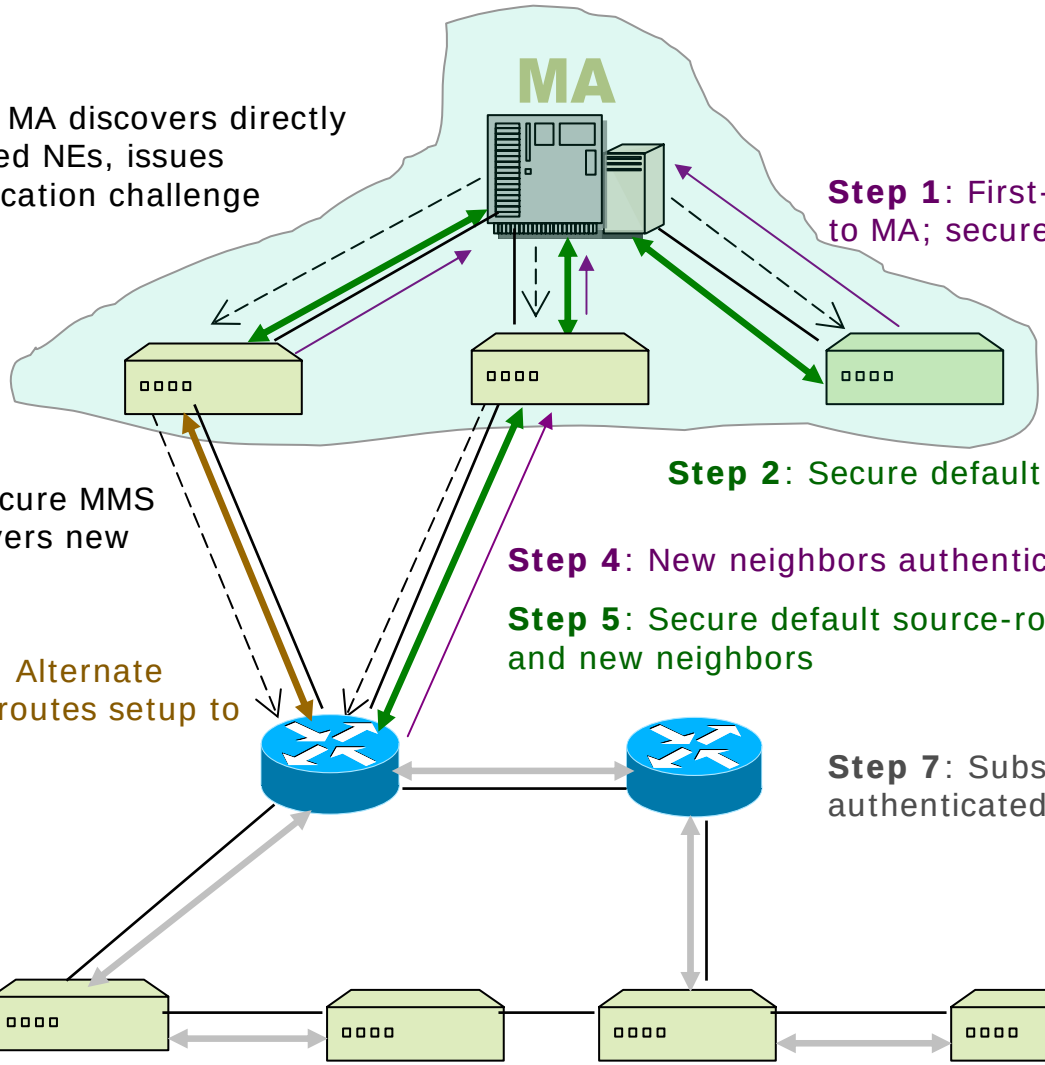
A Meta-Management System (MMS)

- Service on power-up
  - No complex configuration
- Self-contained software
  - Minimizes dependency
- "BIOS" for the network



- Same familiar socket programming interface
  - IP-based management applications still work
- Independent of data-plane services
  - Regular traffic can continue using IP, or not

MMS Authentication & Secure Source Routing



Step 0: MA discovers directly connected NEs, issues authentication challenge

Step 1: First-hop nodes authenticate to MA; secure MMS zone established

Step 2: Secure default source routes are setup to MA

Step 3: Secure MMS zone discovers new neighbors

Step 4: New neighbors authenticate to MA via secure zone

Step 5: Secure default source-routes setup between MA and new neighbors

Step 6: Alternate source-routes setup to MA

Step 7: Subsequent NEs discovered and authenticated recursively

MMS Status

- Prototype system in Linux
  - Linux loadable kernel module
  - Provides a virtual interface (mm0)
  - Detects physical interfaces & discovers neighbors
  - Encryption and authentication of messages
- Attractive alternative to having communication between GMC and CMs via the Internet...

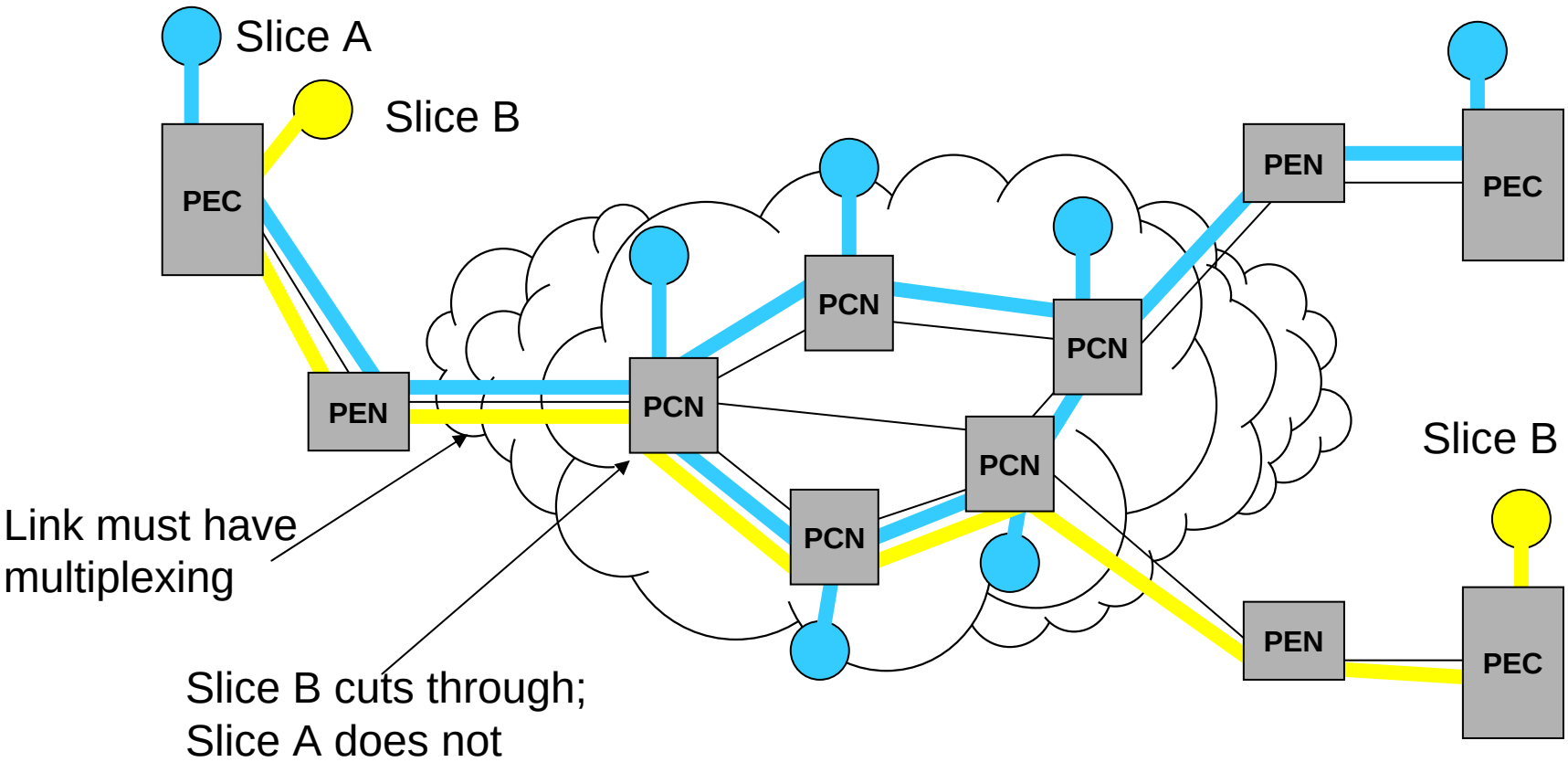
Open Issues

Backbone Issues

- Hardware platform
  - Role of multi-core vs. network processors
  - Right mix of cards in packet processing system
  - Programmability vs. share-ability of processors
  - Monitoring capabilities on high-speed links
- Management software
  - Top-down configuration vs. end-to-end signaling
  - Gateway for interaction with the optical node
  - Division between management aggregate and CM
- Exposing layer 2 and layer 1 failures
  - Explicit notification vs. probing by the substrate

General Issues: Life of a Packet

- Component multiplexes slivers through its VM
- Component must (de)multiplex link traffic
- “Cut through”: no local process in component



General Questions: Packet Headers

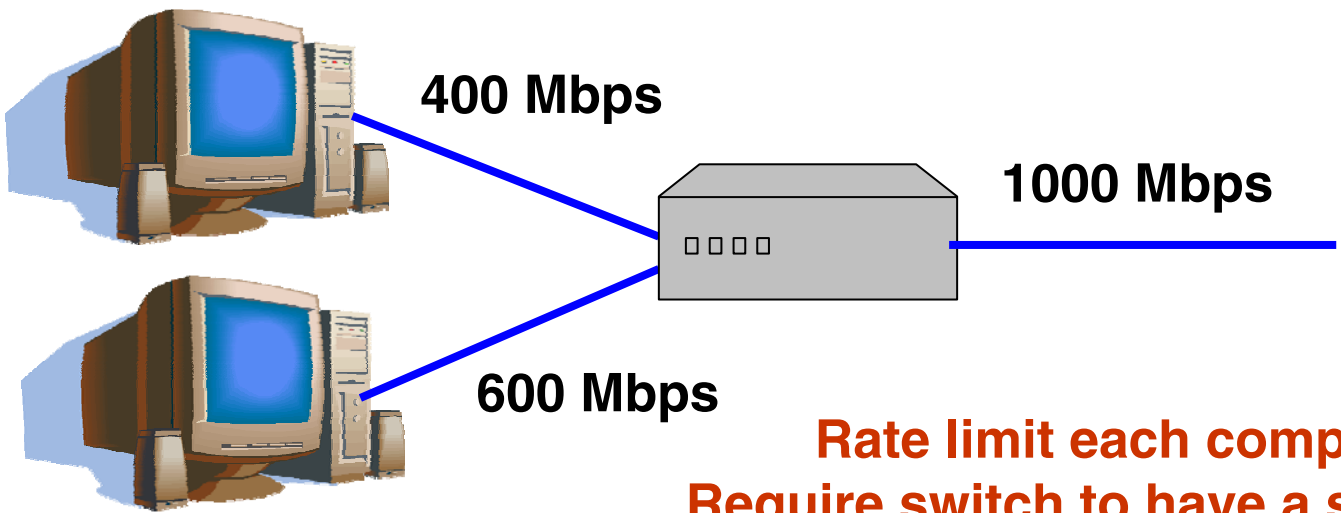
- Common format across all components?
  - Or, allow each link in GENI to use its own?
- Common id for a slice across all hops?
  - Globally unique id vs. link-local ids?
- Existing header format vs. define new one
  - E.g., MPLS tag, IP/port, VLAN tag, ...
- GENI-wide MTU vs. fragmentation in substrate
  - If an MTU, what size?
- One answer: global format, link local, IP address/port, 1500B with no fragmentation

General Issues: Quality-of-Service

- Agree on a set of service models?
  - Best-effort with fair sharing of excess
  - Min bandwidth or max delay guarantee
  - Non-work-conserving for repeatability
- Mechanisms for enforcement
  - Shaping vs. scheduling?
  - At every sliver (VM) or also at cut-through hops?
  - Handling non-QoS capable intermediate devices?
- How much agreement is needed GENI-wide?
  - Vs. how much can we keep under-specified?

General Issues: Non-QoS Capable Devices

- Intermediate devices that don't do QoS
  - Several computers connected to a single link
  - Some best-effort slices, and some guaranteed
- Simple case: rate limit each computer
  - Slivers only capitalize on excess b/w on same CPU
  - No overbooking, so no overload of the link



Rate limit each computer?
Require switch to have a scheduler?