# Evaluation Strategies

Nick Feamster
CS 7260
February 26, 2007

# Evaluation Strategies

- Many ways to evaluate new protocols, systems, implementations
  - Mathematical analysis
  - Simulation (ns, SSFNet, etc.)
  - Emulation (emulab)
  - Trace-driven evaluation
  - Wide-area deployment (VINI)
- Interplay between these areas is not obvious!
- Various tradeoffs in “realism”, control, etc.
- A combination may be appropriate

# Why Network Simulation?

- Can capture complexity that analytical models miss
- Protocol validation
  - Quantitative results
  - Exploration of dynamics
- Controlled experimental conditions
- Low cost/barrier to entry
  - Time
  - Collaboration
  - Complexity

# Simulation: *ns*

- *ns*:discrete-event network simulator for Internet systems
  - protocol design, large scale systems studies, prototyping, education

- Why *ns?*
  - Protocols: TCP, UDP, HTTP, etc.
  - Traffic Models: Web Traffic, CBR,
  - Topology Generation tools
  - Visualization tools

# Step 1: Topology

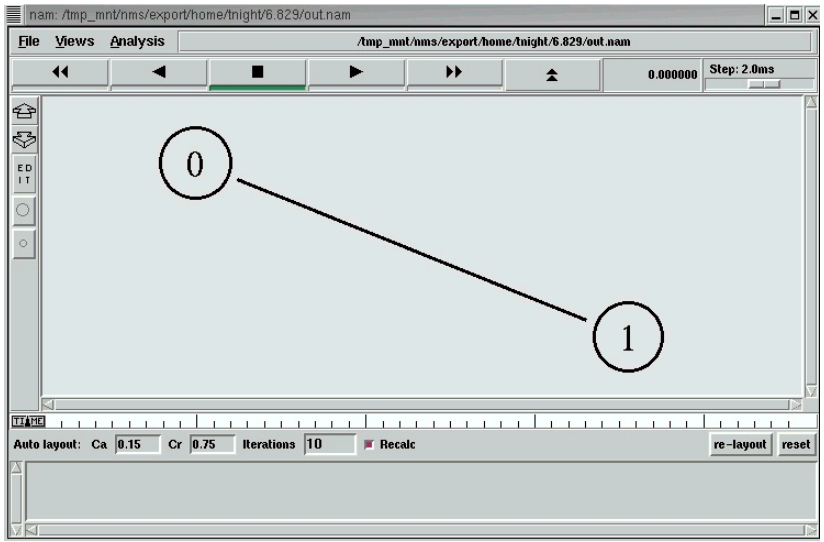
```
# Create simulation object
set ns [new Simulator]

# Ask ns for nodes
set n0 [$ns node]
set n1 [$ns node]

# Create a duplex link b/w n0 & n1
$ns duplex-link $n0 $n1 1Mb 10ms DropTail


# Schedule End
$ns at 5.0 "exit 0"

# Run Simulation
$ns run
```



# Step 2: Attaching Agents

- **Purpose:** Transport connections between nodes
- Various types: TCP, UDP, etc.

**UDP**

```
# Create a UDP agent
set udp1 [new Agent/UDP]

# Create a Null agent
set sink1 [new Agent/Null]

# Attach agent udp1 to node n0
$ns attach-agent $n0 $udp1

# Attach agent sink1 to node n1
$ns attach-agent $n1 $sink1

# Connect the agents
$ns connect $udp1 $sink1
```

**TCP**

```
# Create a TCP agent
set tcp1 [new Agent/TCP]

# Create a Null agent
set sink1 [new Agent/TCPSink]

# Attach agent tcp1 to node n0
$ns attach-agent $n0 $tcp1

# Attach agent sink1 to node n1
$ns attach-agent $n1 $sink1

# Connect the agents
$ns connect $tcp1 $sink1
```

# Step 3: Creating Traffic

- **Purpose:** Send traffic over links/transport

```
# Create Source
set cbr1 [new Application/Traffic/CBR]

# Configure Source
$cbr1 set packetSize_ 500 $cbr1 set
interval_ 0.005

# Attach source to agent
$cbr1 attach-agent $udp1

# Schedule cbr on
$ns at 0.5 "$cbr1 start"

# Schedule cbr off
$ns at 4.5 "$cbr1 stop"
```

# Simulation: Advantages and Disadvantages

## Advantages

- Ease of use
- Often possible to achieve large scale (federation, etc.)
- Low cost in time, money, etc.
- Many “accepted” models available

## Disadvantages

- Models may be (and have previously been shown to be) incorrect
- Doesn’t run actual protocols, software implementations, etc.
- Parameter exploration creep

# Crisis of Credibility

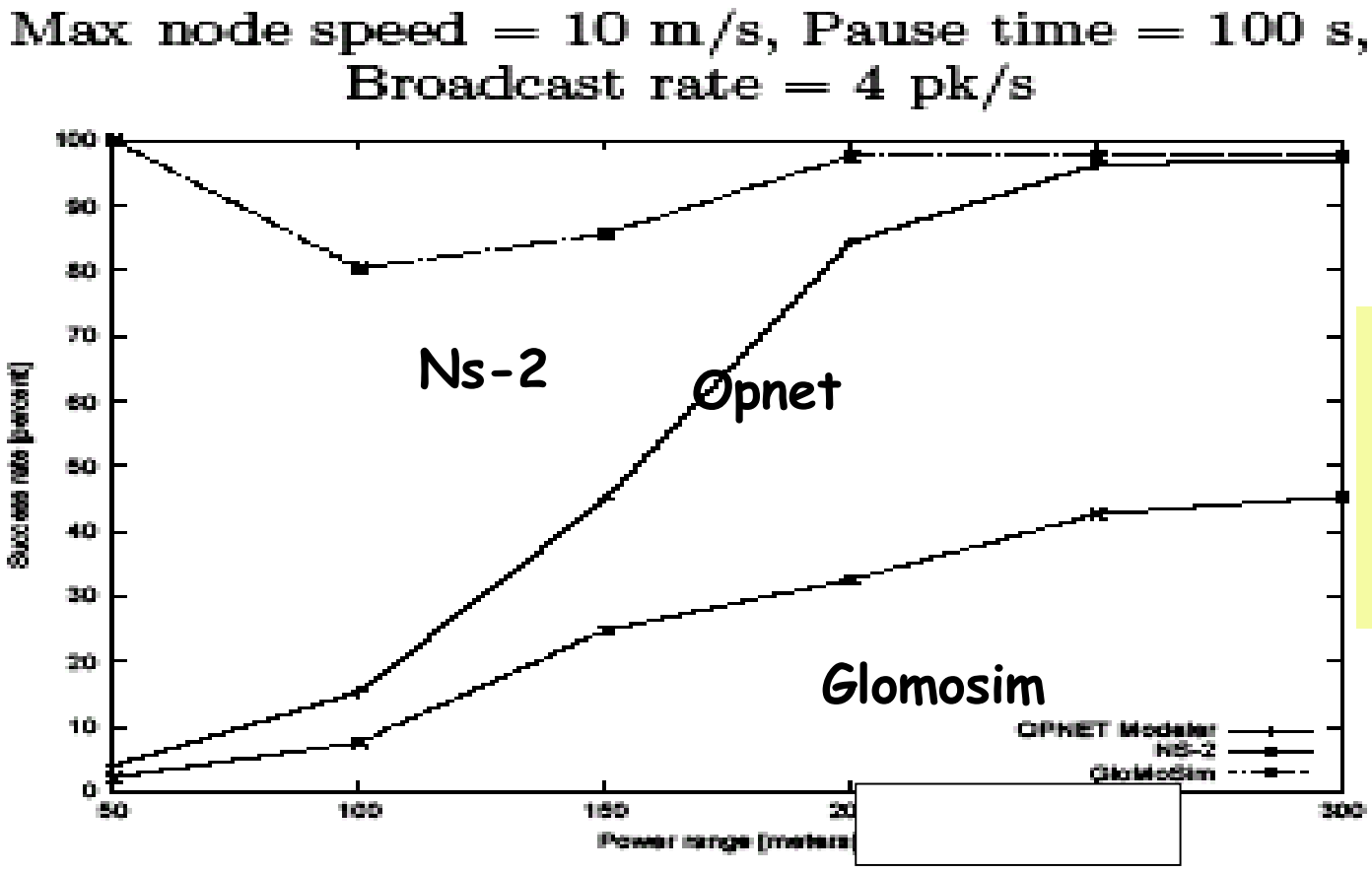



Figure 3: Success rate vs Power range

From: Cavin, Sasson and Schiper – On the accuracy of MANET Simulators

# Case Study: Internet

- Heterogeneity
  - Link media (fiber, copper, wireless, etc.)
  - Link rates
  - Transport protocol *implementations*
- Scale
  - When is it OK to draw conclusions from small-scale experiments?
- Drastic rates of change

"Difficulties in Simulating the Internet", *IEEE/ACM ToN*, August 2001

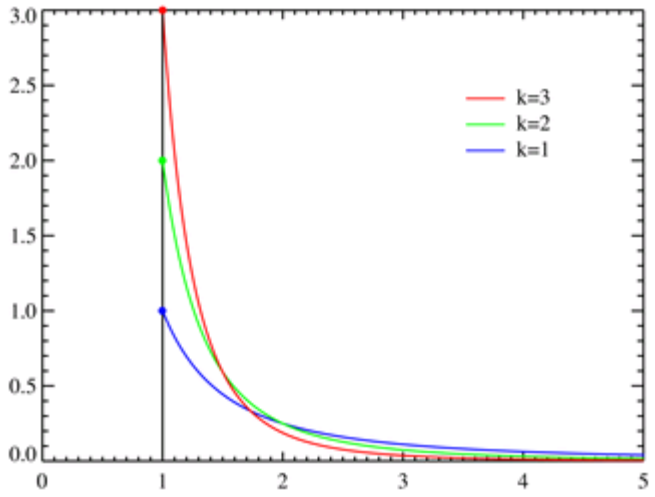
# Coping Strategy: Invariants

- Diurnal traffic patterns
- Self-similarity/Long-range dependence
- Heavy-tailed distributions

$$\Pr[X > x] \sim x^{-\alpha} \text{ as } x \to \infty, \qquad 0 < \alpha < 2.$$

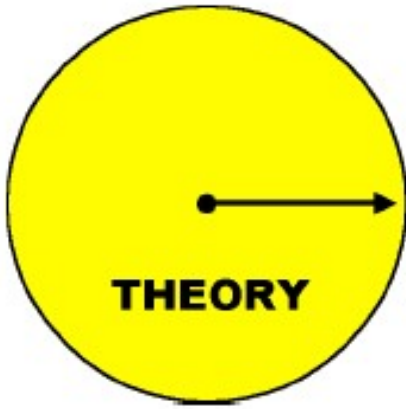
- Speed of light

**Pareto Distributions**

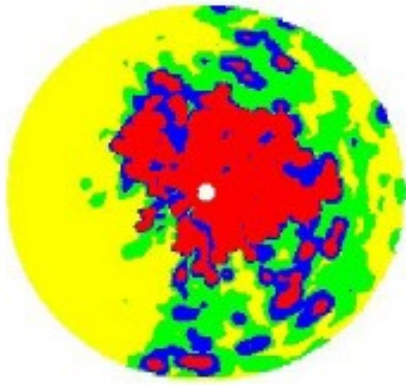


# Case Study: Wireless

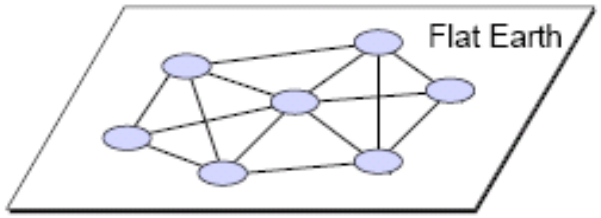
- The earth is not flat
- Radio transmission range is not circular
- Radios have unequal ranges
- Communication is asymmetric
- Reachability does not imply perfect communication
- Signal strength is not only a function of distance

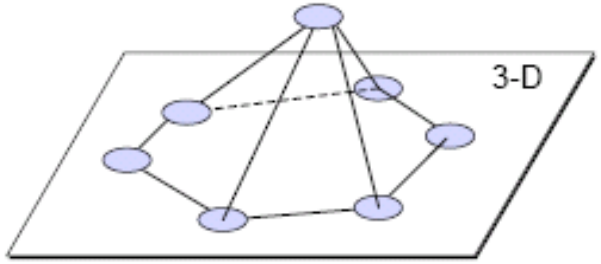


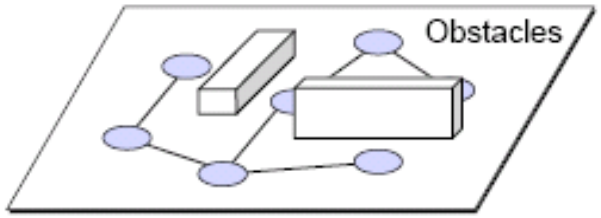

Typical theoretical model



Source: Comgate Engineering
http://www.comgate.com/ntdsign/wireless.l.....




"Mistaken Axioms of Wireless Networking Research", Dartmouth TR

# Emulation: Emulab

http://www.emulab.net/

# Why?

- “We evaluated our system on five nodes.”
  - job talk from university with 300-node cluster
- “We evaluated our Web proxy design with 10 clients on 100Mbit ethernet.”
- “Simulation results indicate ...”
- “Memory and CPU demands on the individual nodes were not measured, but we believe will be modest.”
- “The authors ignore interrupt handling overhead in their evaluation, which likely dominates all other costs.”
- “Resource control remains an open problem.”

# Emulab Design Features

- Allow experimenter **complete control**
- … but provide *fast* tools for common cases
  - OS's, disk loading, state mgmt tools, IP, traffic generation, batch, ...

- Virtualization
  - of *all* experimenter-visible resources
  - node names, network interface names, network addresses
  - Allows swapin/swapout

# Design Aspects (cont'd)

- Flexible, extensible, powerful allocation algorithm
- Persistent state maintenance:
  - none on nodes
  - all in database
  - leverage node boot time: only known state!
- Separate control network
- Familiar, powerful, extensible configuration language: *ns*

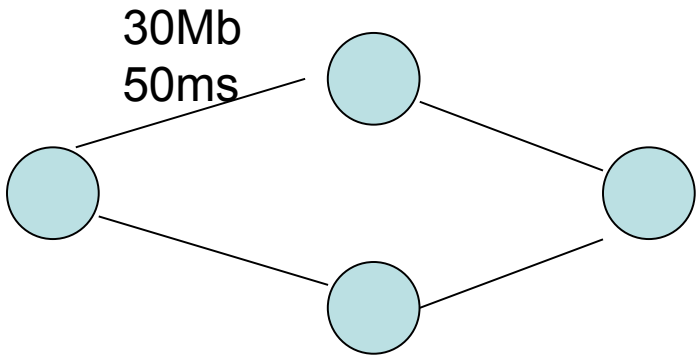
# More Unique Characteristics

- Capture of low-level node behavior such as interrupt load and memory bandwidth
- User-replaceable node OS software
- User-configurable physical link topology
  - User-configurable control of “physical” characteristics:
    - shaping of link latency/bandwidth/drops/errors (via invisibly interposed “shaping nodes”),
    - router processing power,
    - buffer space, …
- Configurable by external researchers, including node power cycling

# Example Topology and Configuration

```
set ns [new Simulator]
source tb_compat.tcl

set nodeA [$ns node]
set nodeB [$ns node]
set nodeC [$ns node]
set nodeD [$ns node]
```




```
set link0 [$ns duplex-link $nodeA $nodeB 30Mb 50ms DropTail]
set link1 [$ns duplex-link $nodeA $nodeC 30Mb 50ms DropTail]
set link2 [$ns duplex-link $nodeC $nodeD 30Mb 50ms DropTail]
set link3 [$ns duplex-link $nodeB $nodeD 30Mb 50ms DropTail]

$ns rtproto Static
$ns run
```

# Demonstration

# What Is It Not Good For?

- Packet-level expts. across many nodes
  - Clock synchronization good, but not perfect
  - Non-determinism in the real world
- Experiments that require real routers
  - All nodes are PCs
    - But, we can use a few different queuing strategies
    - And, you can reprogram them all you want
- Experiments that require gigabit links
  - None yet, but we hope to add some
- Experiments that need 1000s of links/nodes
  - ModelNet, coming soon, will help

# Challenges for Emulation and Wide-Area Deployment

- **Mapping and embedding**
  - Resource management
  - Scheduling experiments
- Scaling
- Validation
- Security
- Artifact detection and control
- User interface issues

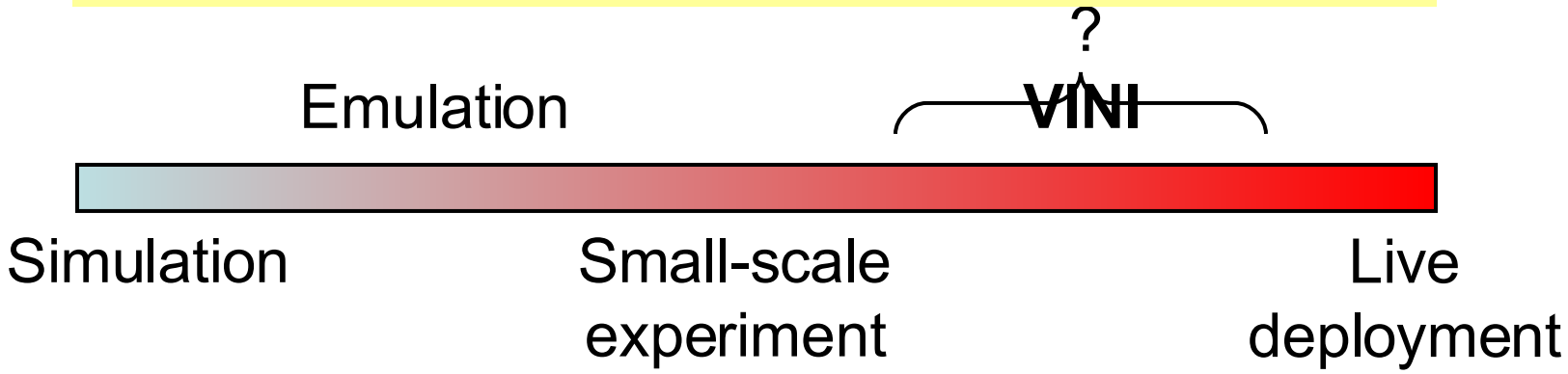
# Network Embedding Problem

- **Given:** virtual network and physical network
  - Topology, constraints, etc.
- **Problem:** find the appropriate mapping onto available physical resources (nodes and edges)

- Many possible formulations
  - Specific nodes mapping to certain physical nodes
  - Generic requirements: “three diverse paths from SF to LA with 100 MBps throughput”
  - Traffic awareness, dynamic remapping, etc.

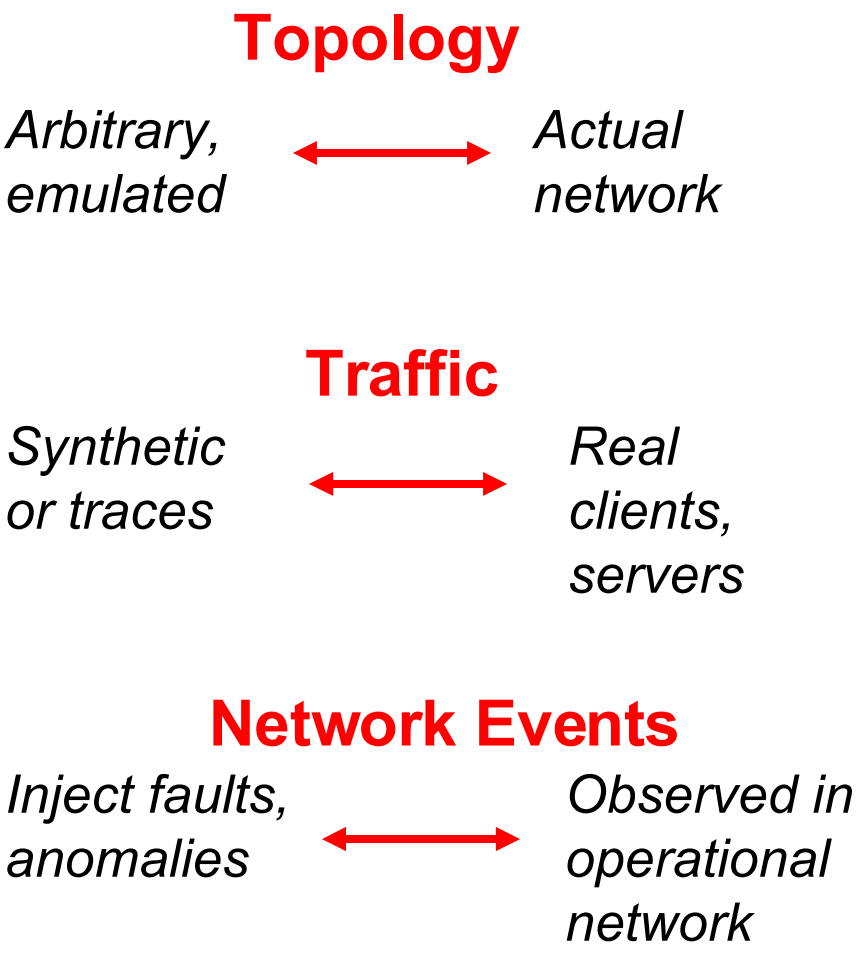
# VINI Overview

**Bridge the gap between "lab experiments" and live experiments at scale.**




- Runs real routing software
- Exposes realistic network conditions
- Gives control over network events
- Carries traffic on behalf of real users
- Is shared among many experiments

# Goal: Control and Realism




- **Control**
  - *Reproduce results*
  - Methodically change or relax constraints
- **Realism**
  - Long-running services attract real users
  - Connectivity to real Internet
  - Forward high traffic volumes (Gb/s)
  - Handle unexpected events

# Network Virtualization: Characteristics

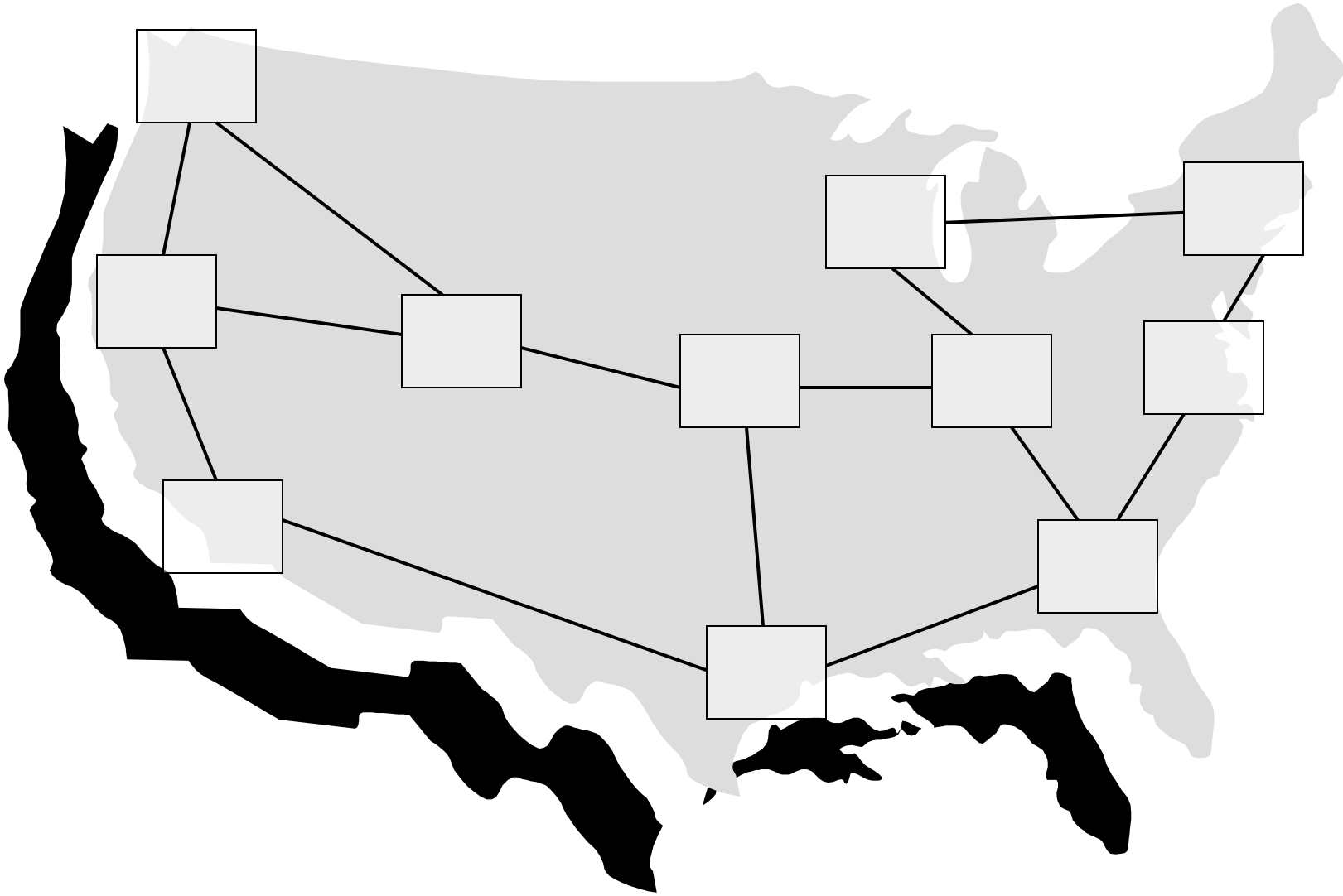
## Sharing

- Multiple logical routers on a single platform
- Resource isolation in CPU, memory, bandwidth, forwarding tables, …

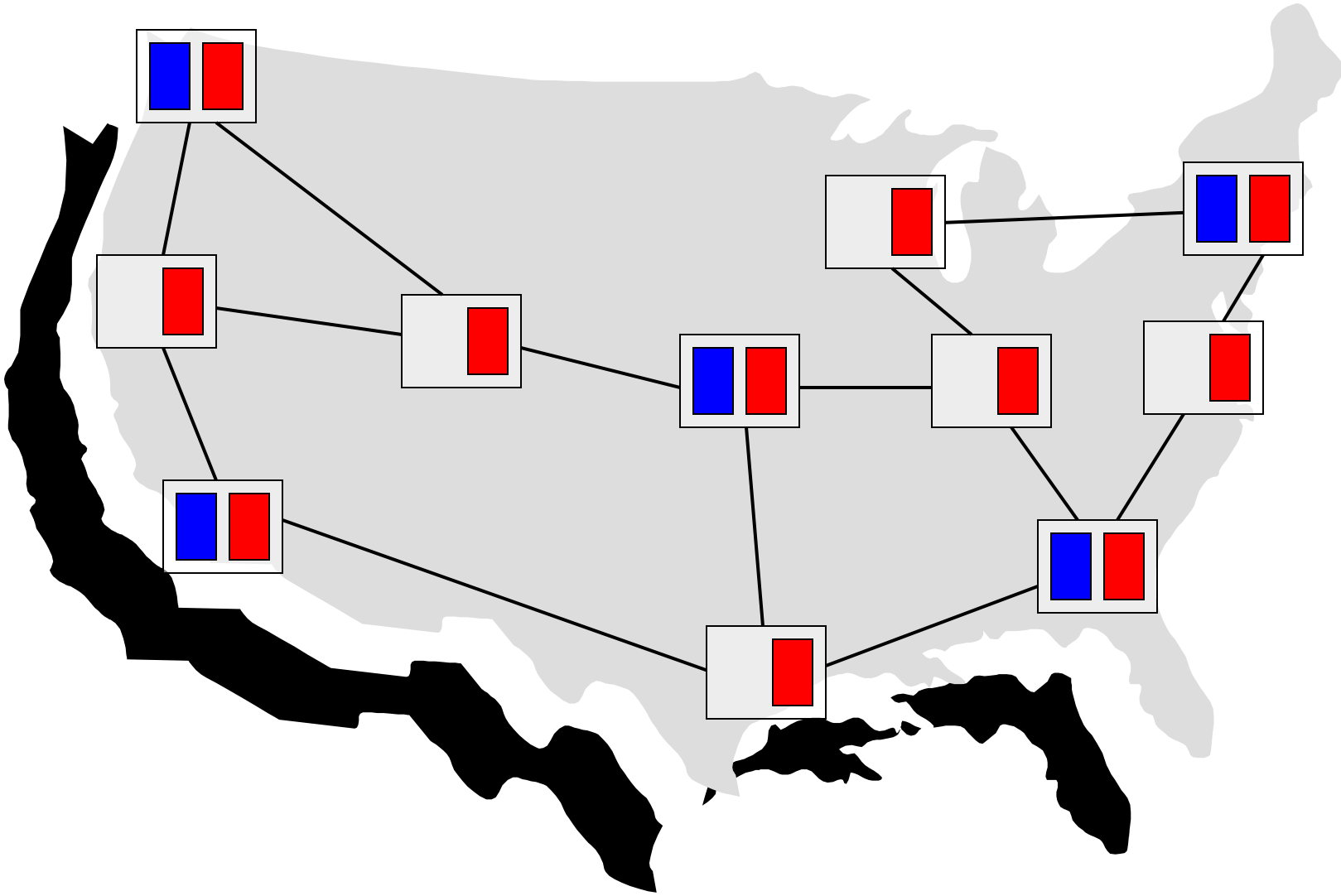
## Customizability

- Customizable routing and forwarding software
- General-purpose CPUs for the control plane
- Network processors and FPGAs for data plane

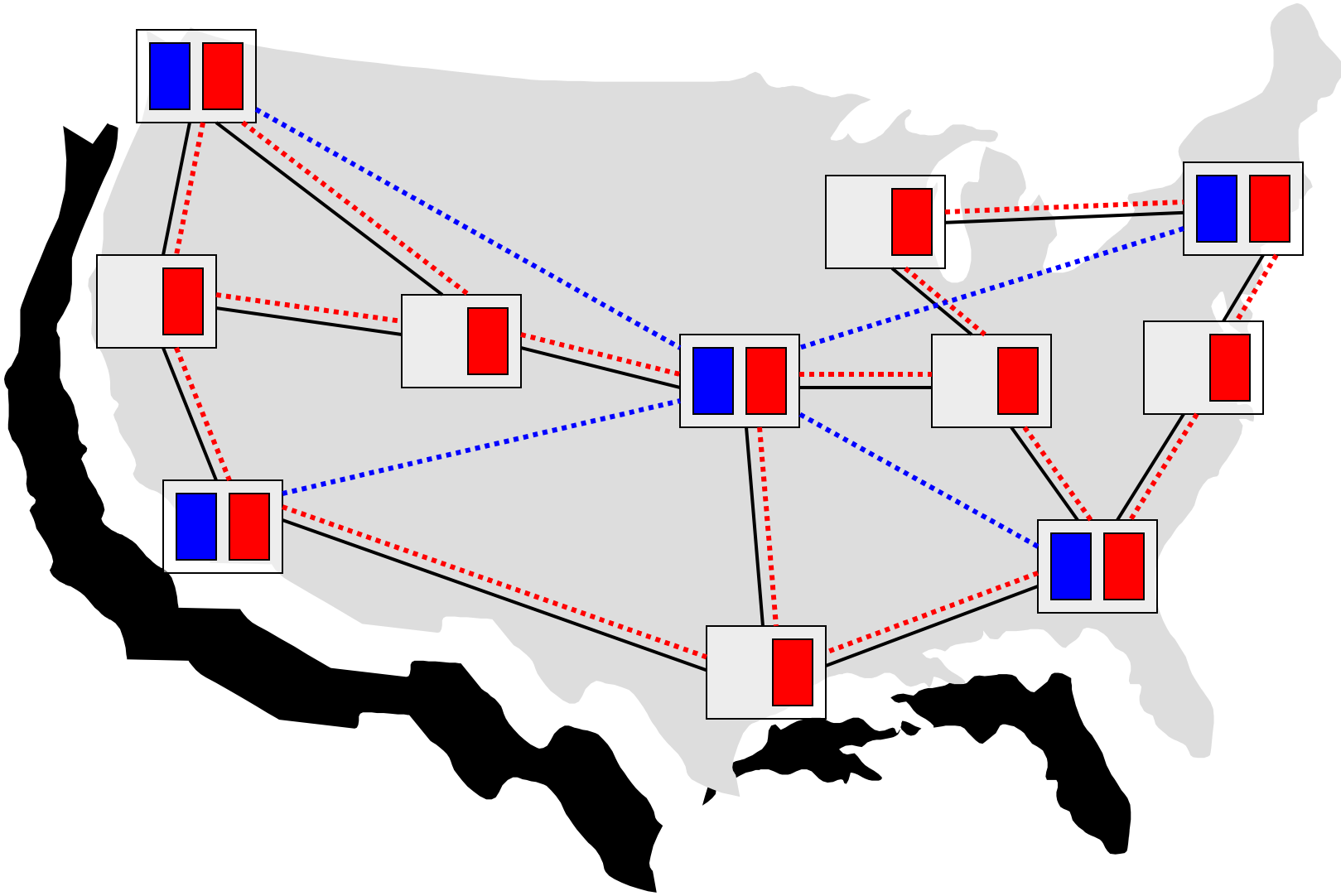
# Fixed Physical Infrastructure



# Shared By Many Parties



# Supports Arbitrary Virtual Topologies



# Why Is This Difficult?

- Creation of virtual nodes
  - Sharing of resources
  - Creating the appearance of multiple interfaces
  - Arbitrary software
- Creation of virtual links
  - Expose underlying failures of links
  - Controlled link failures
  - Arbitrary forwarding paradigms
- Embedding virtual topologies
  - Support for simultaneous virtual experiments
  - Must map onto available resources, account, etc.

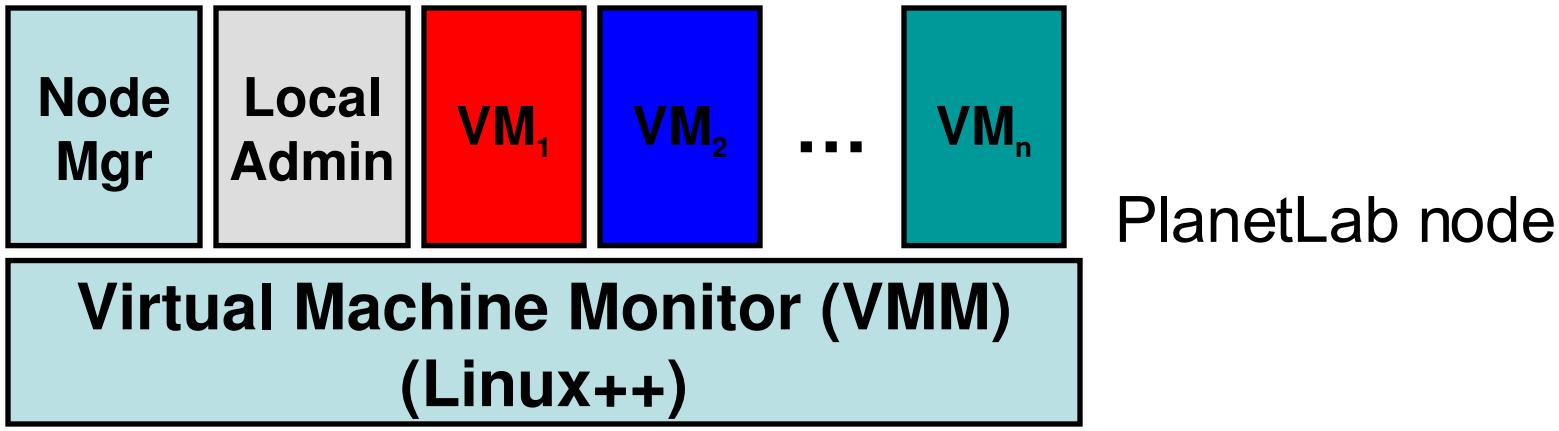
# PL-VINI: Prototype on PlanetLab

- **First experiment:** *Internet In A Slice*
  - XORP open-source routing protocol suite
  - Click modular router

- Expose issues that VINI must address
  - Unmodified routing (and other) software on a virtual topology
  - Forwarding packets at line speed
  - Illusion of dedicated hardware
  - Injection of faults and other events

# PL-VINI: Prototype on PlanetLab

- PlanetLab: testbed for planetary-scale services
- Simultaneous experiments in separate VMs
  - Each has “root” in its own VM, can customize
- Can reserve CPU, network capacity per VM



# Internet In A Slice

**XORP**

- Run OSPF
- Configure FIB

**Click**

- FIB
- Tunnels
- Inject faults

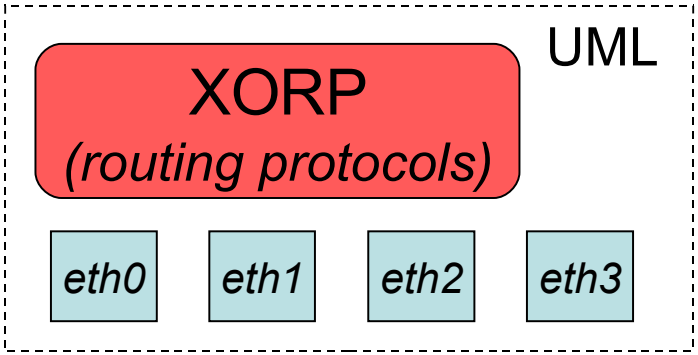
**OpenVPN & NAT**

- Connect clients and servers

# XORP: Control Plane

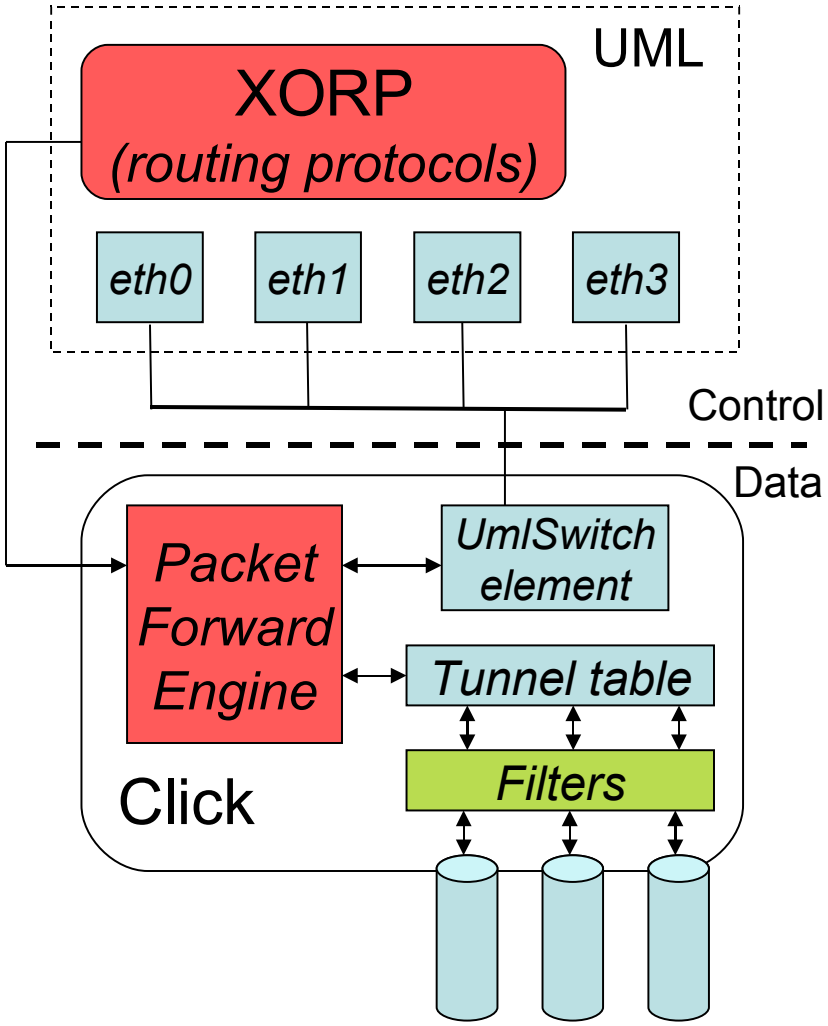

- BGP, OSPF, RIP, PIM-SM, IGMP/MLD
- **Goal: run real routing protocols on virtual network topologies**

# User-Mode Linux: Environment



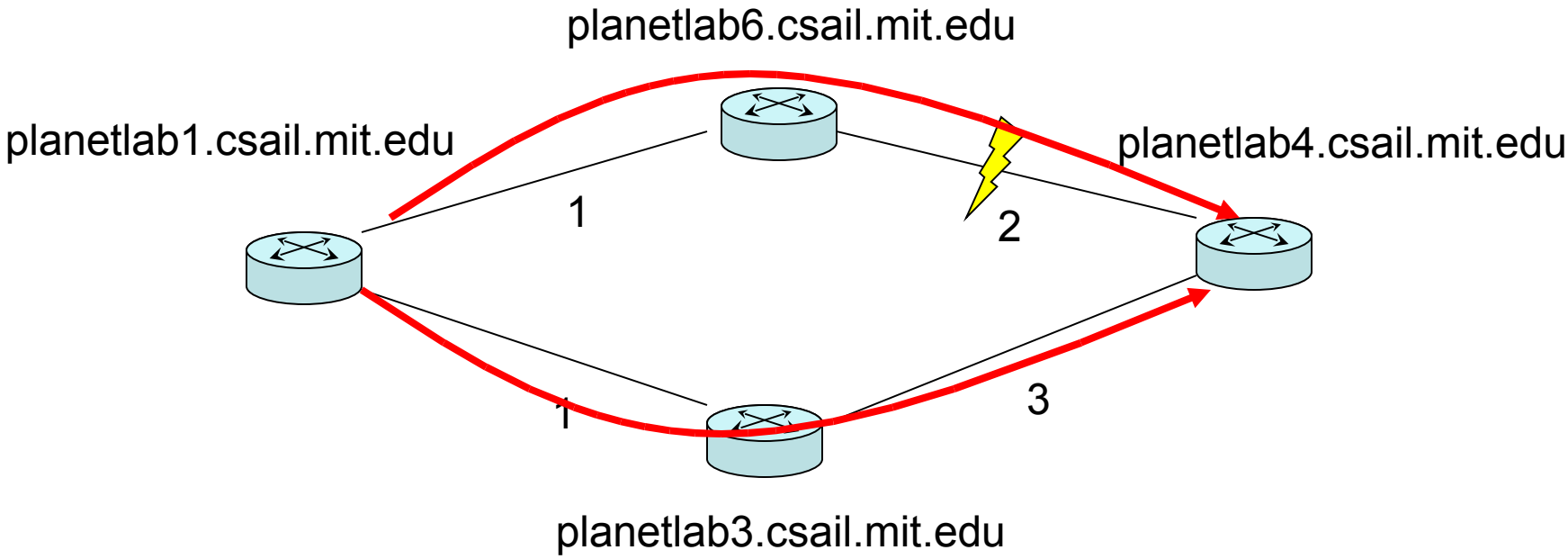

- PlanetLab limitation:
  - Slice cannot create new interfaces
- Run routing software in UML environment
- Create virtual network interfaces in UML
- **Challenge:** Map these interfaces to the right tunnels

# Click: Data Plane




- Performance
  - Avoid UML overhead
  - Move to kernel, FPGA
- Interfaces ⇨ tunnels
  - Click UDP tunnels correspond to UML network interfaces
- Filters
  - “Fail a link” by blocking packets at tunnel

# Demonstration



# Questions to Ask