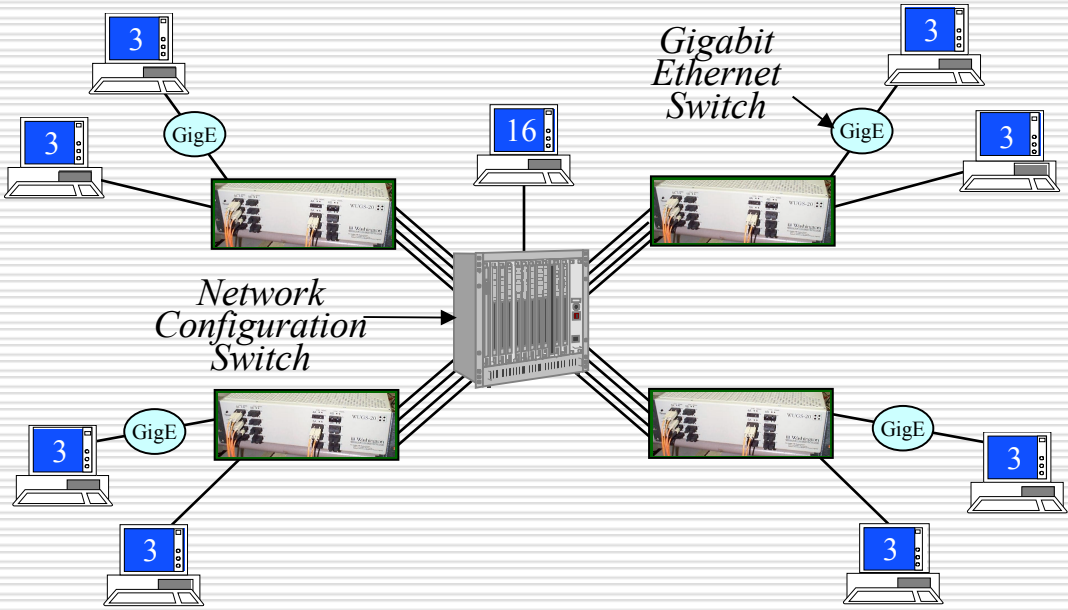
The Open Network Lab

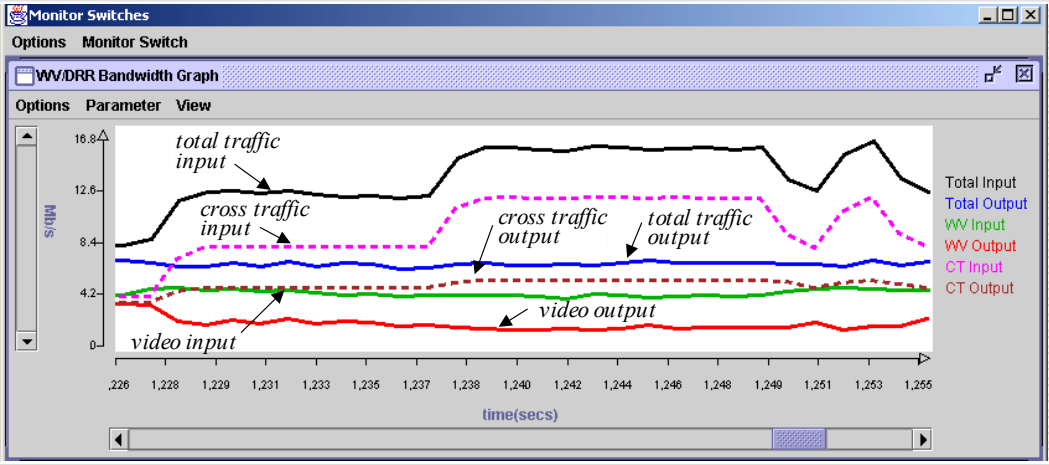
Charlie Wiseman
Applied Research Laboratory
Computer Science and Engineering Department
cgw1@arl.wustl.edu

http://www.onl.wustl.edu (ONL)

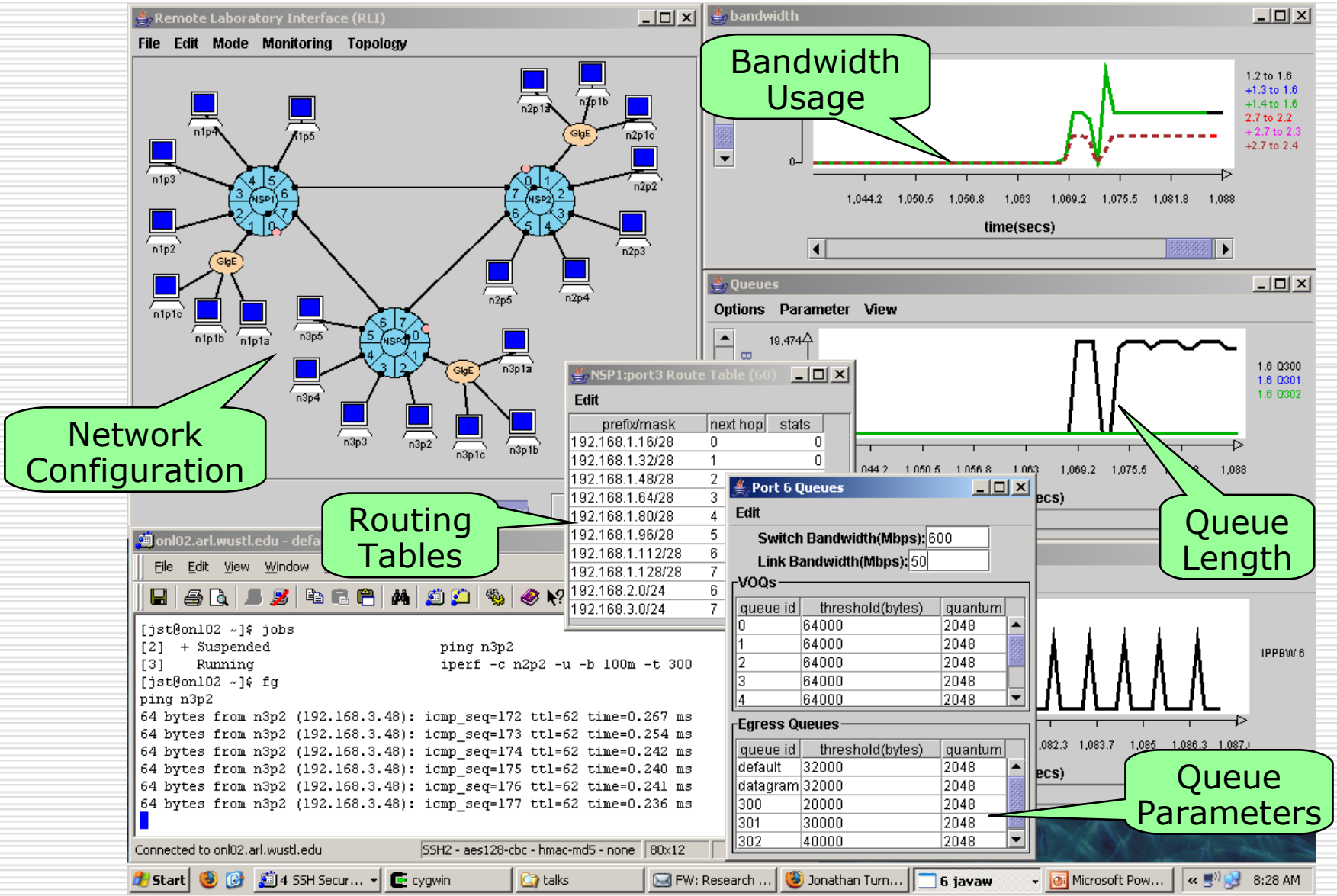
ONL Lab Overview

- Gigabit routers
  - » easily configured thru ***Remote Lab Interface***
  - » embedded processors for adding new features
- PCs serve as hosts
  - » half on shared subnets
- Net configuration switch
  - » link routers in virtual topologies
  - » traffic generation
- Tools for configuration and collecting results
  - » monitoring traffic
  - » data capture and playback
- Open source
  - » all hw & sw sources on web





Sample ONL Session



People Who Make it Happen

Jon Turner
Principle
Investigator

Ken Wong
Admin, Web site,
Dist. Sched.

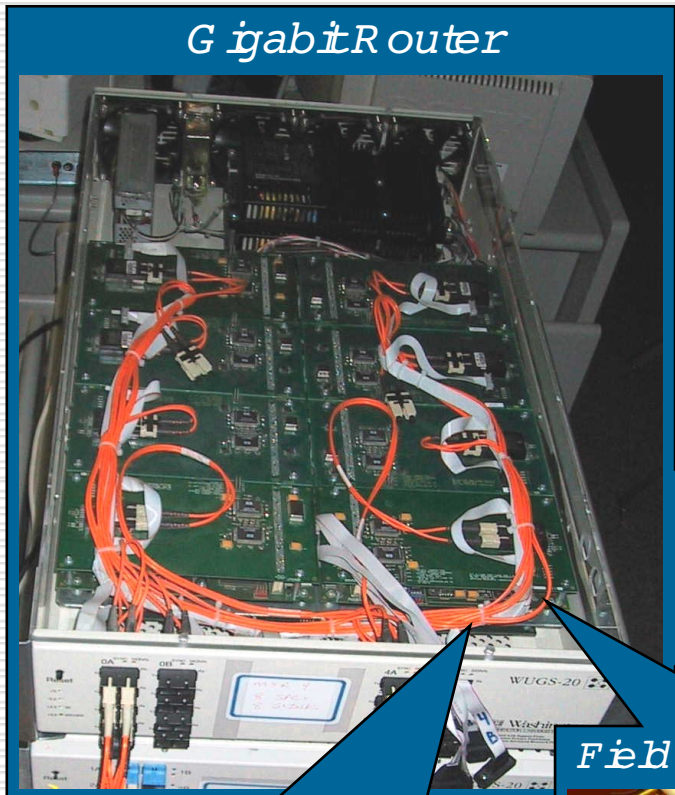
Jyoti Parwatikar
RLI, Software
development

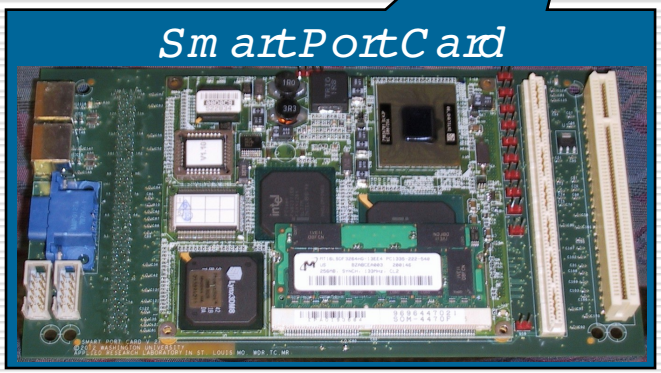
Charlie Wiseman
Web site, Ops
Dist. Sched.

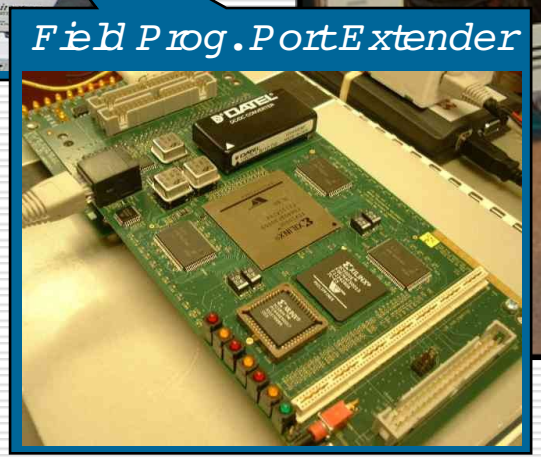
Fred Kuhns
SPC software
FPX hardware

John Dehart
FPX hardware
System integration

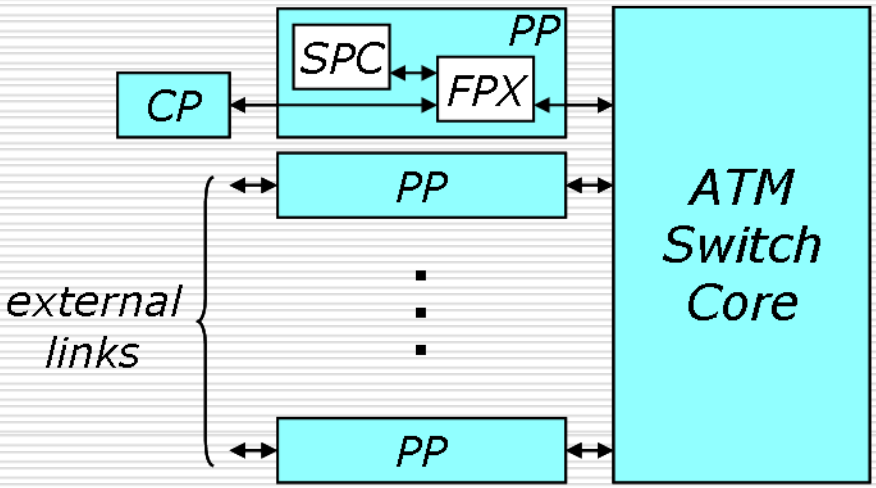
ONL Hardware

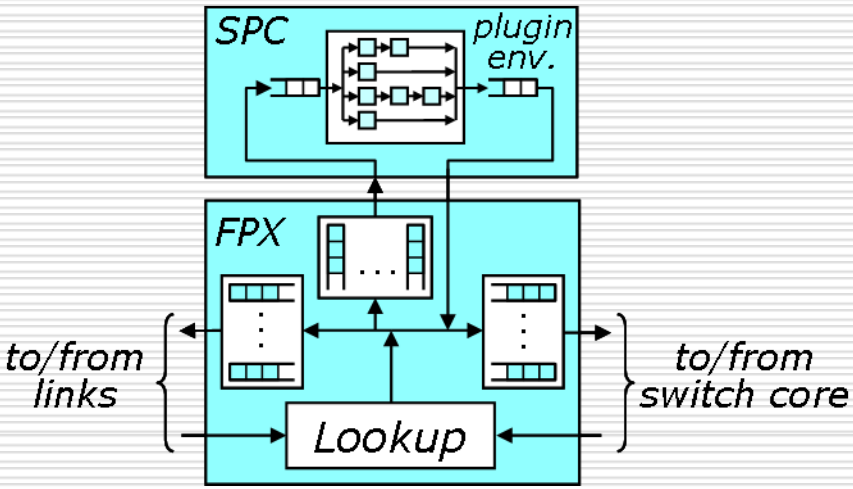






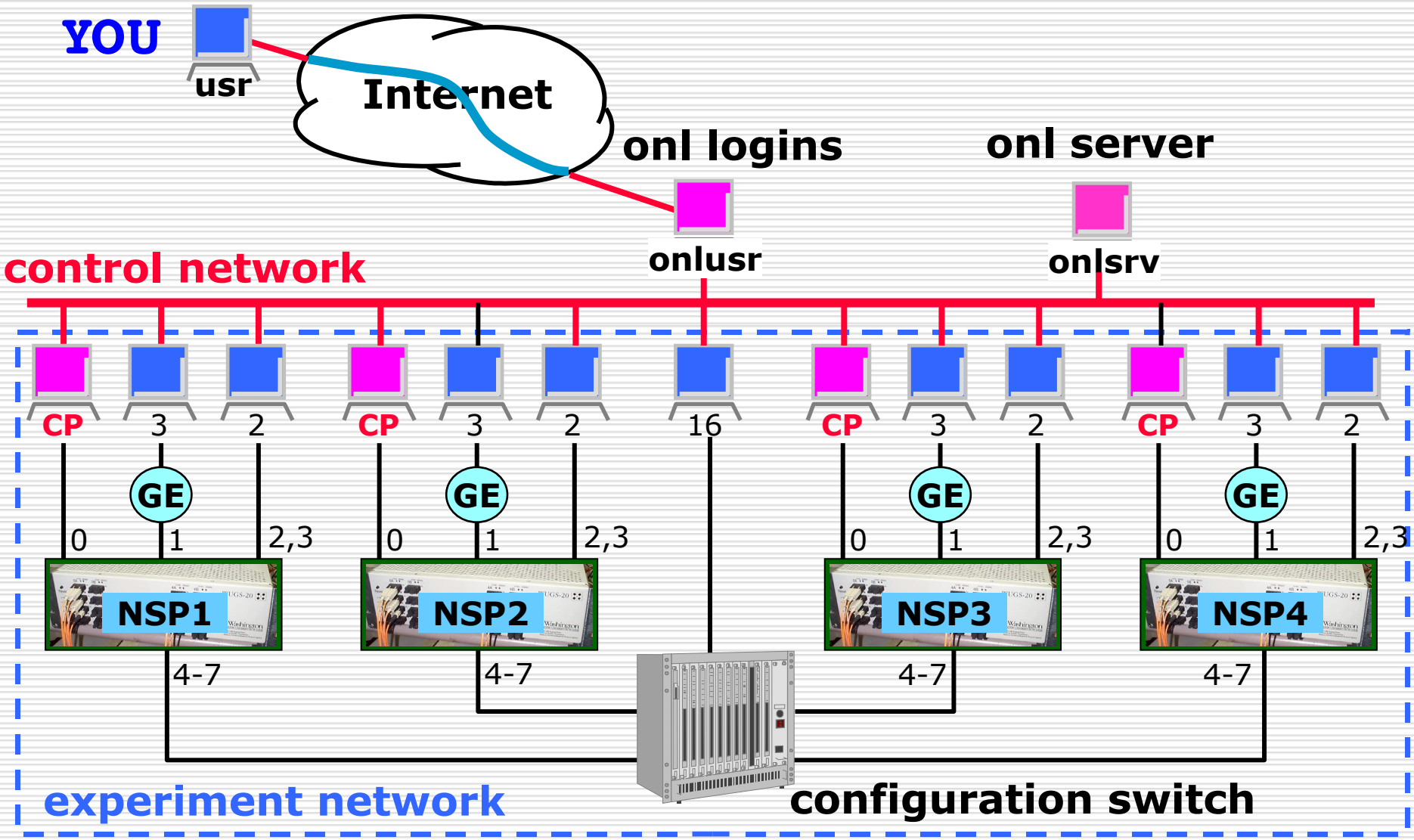
Basic NSP Architecture



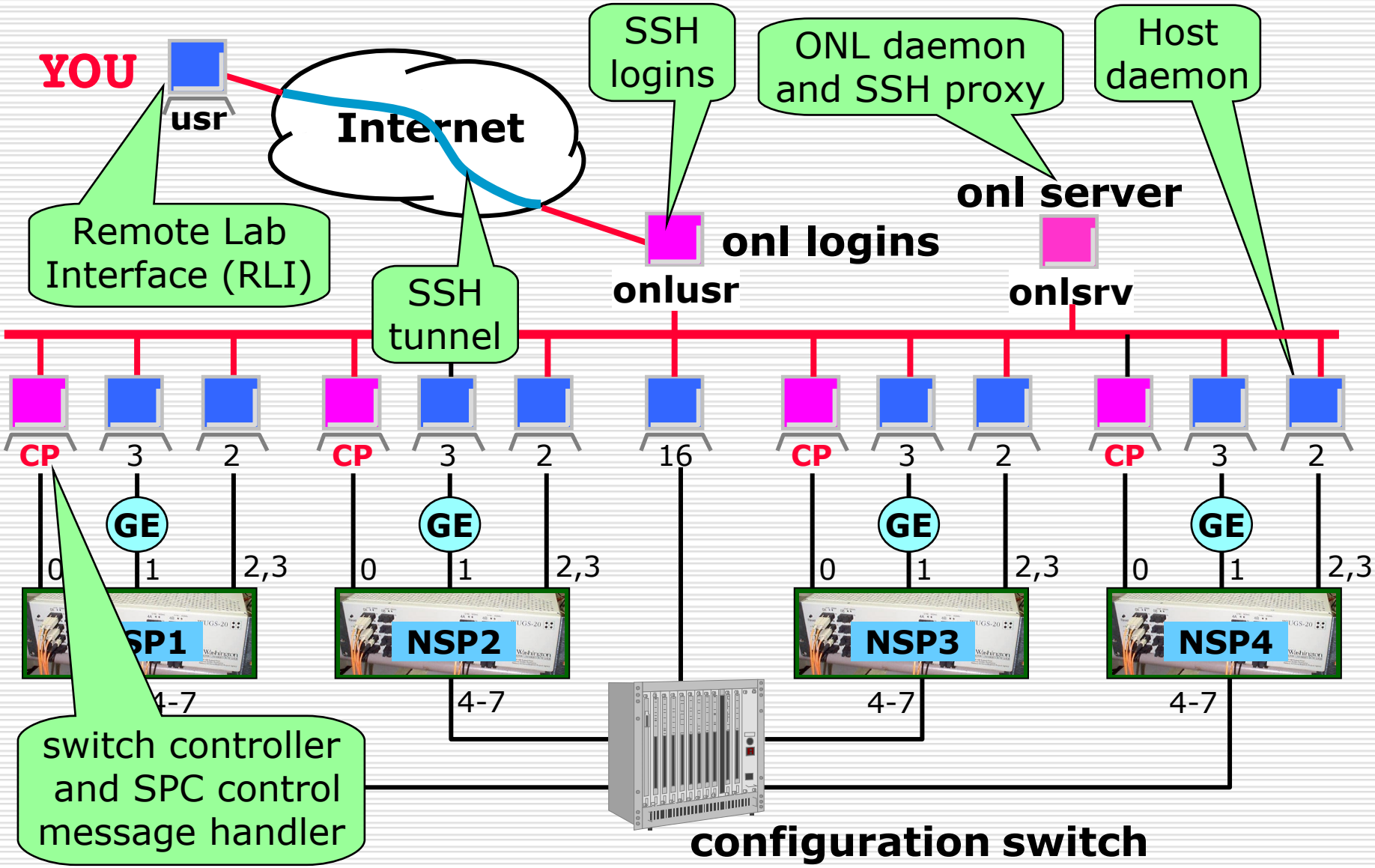


- CP = control processor
- PP = port processor
- SPC = smart port card
- FPX = field programmable port extender

Testbed Organization



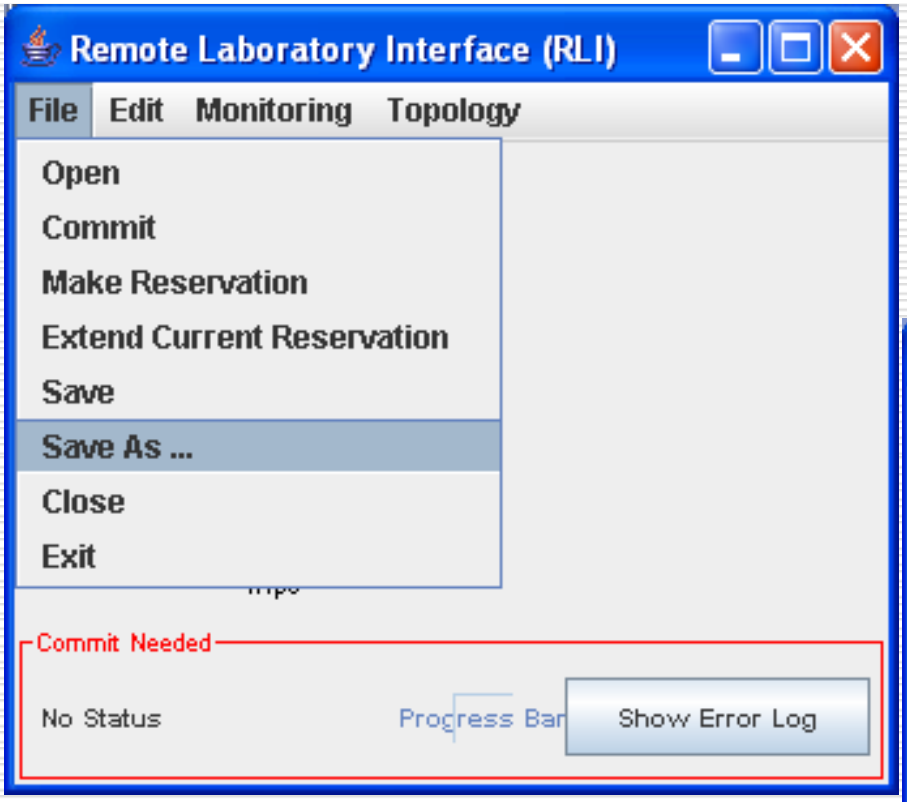
Major Software Components



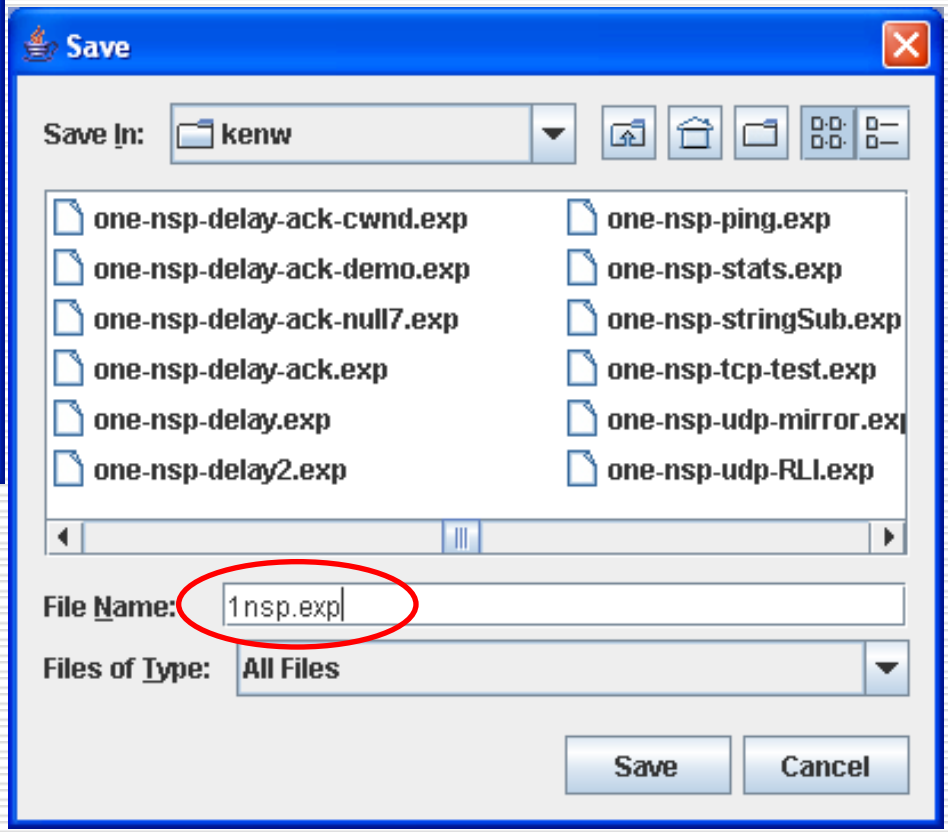
Demo 1

- Topology
- Add cluster
- Add hosts and links
- Generate default routes
- Spin handle and port 0
- Commit

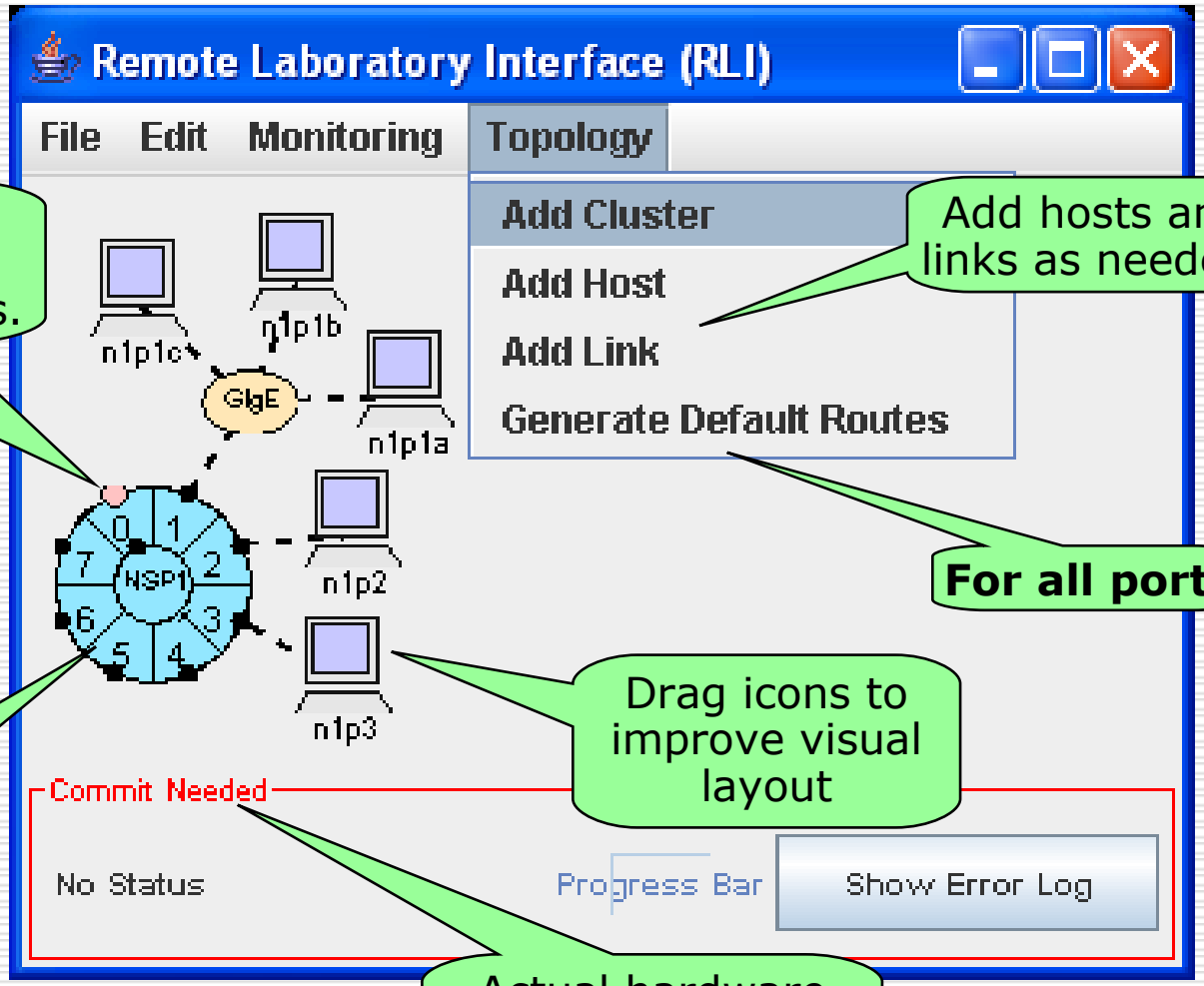
Save Configuration



You do NOT need to be connected to the ONL testbed until you commit !



Configuring Topology



Port 0 used for **Control Processor**. Spin handle rotates ports.

Add hosts and links as needed.

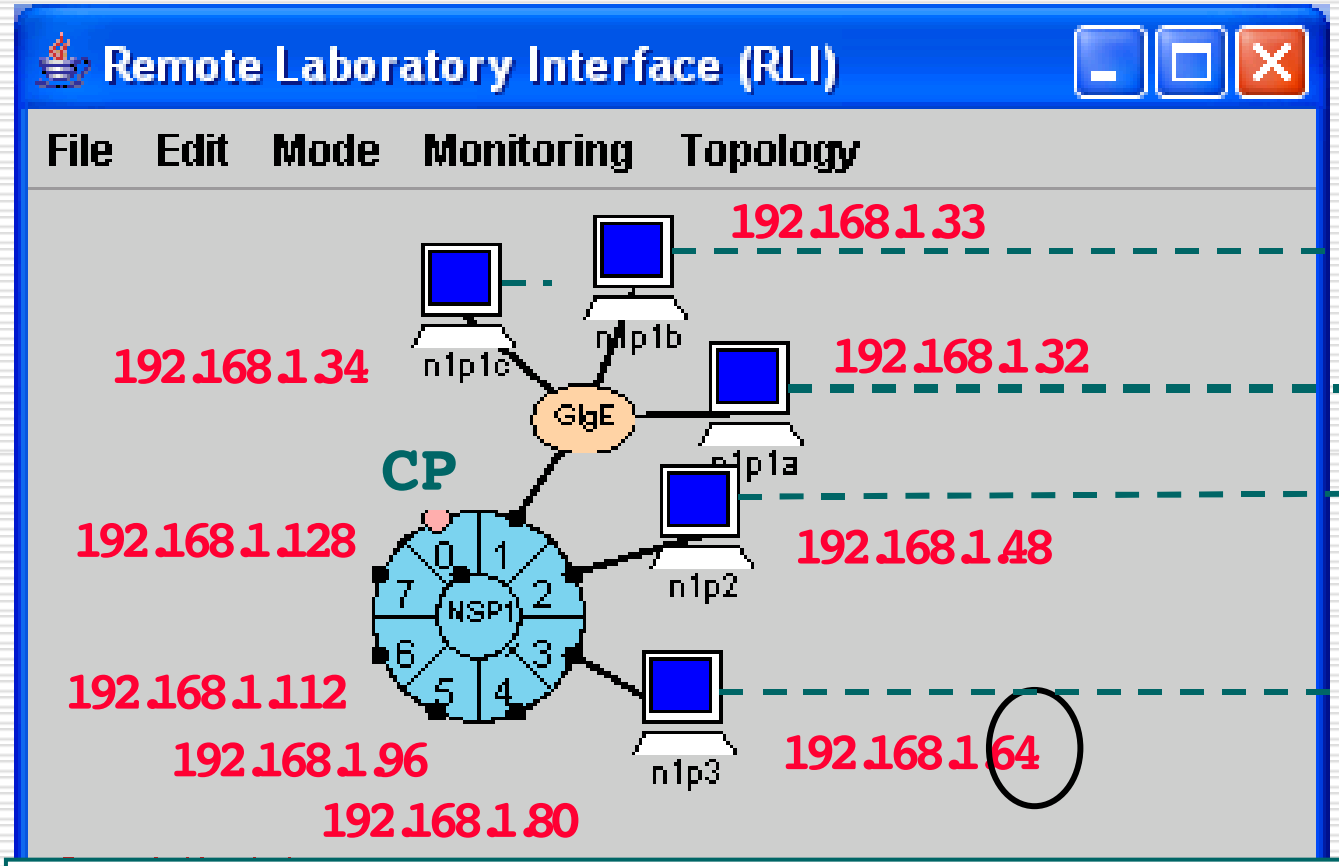
For all ports

Drag icons to improve visual layout

Cluster includes:
- NSP router,
- GE switch,
- fixed set of hosts

Actual hardware **NOT** been allocated until after commit

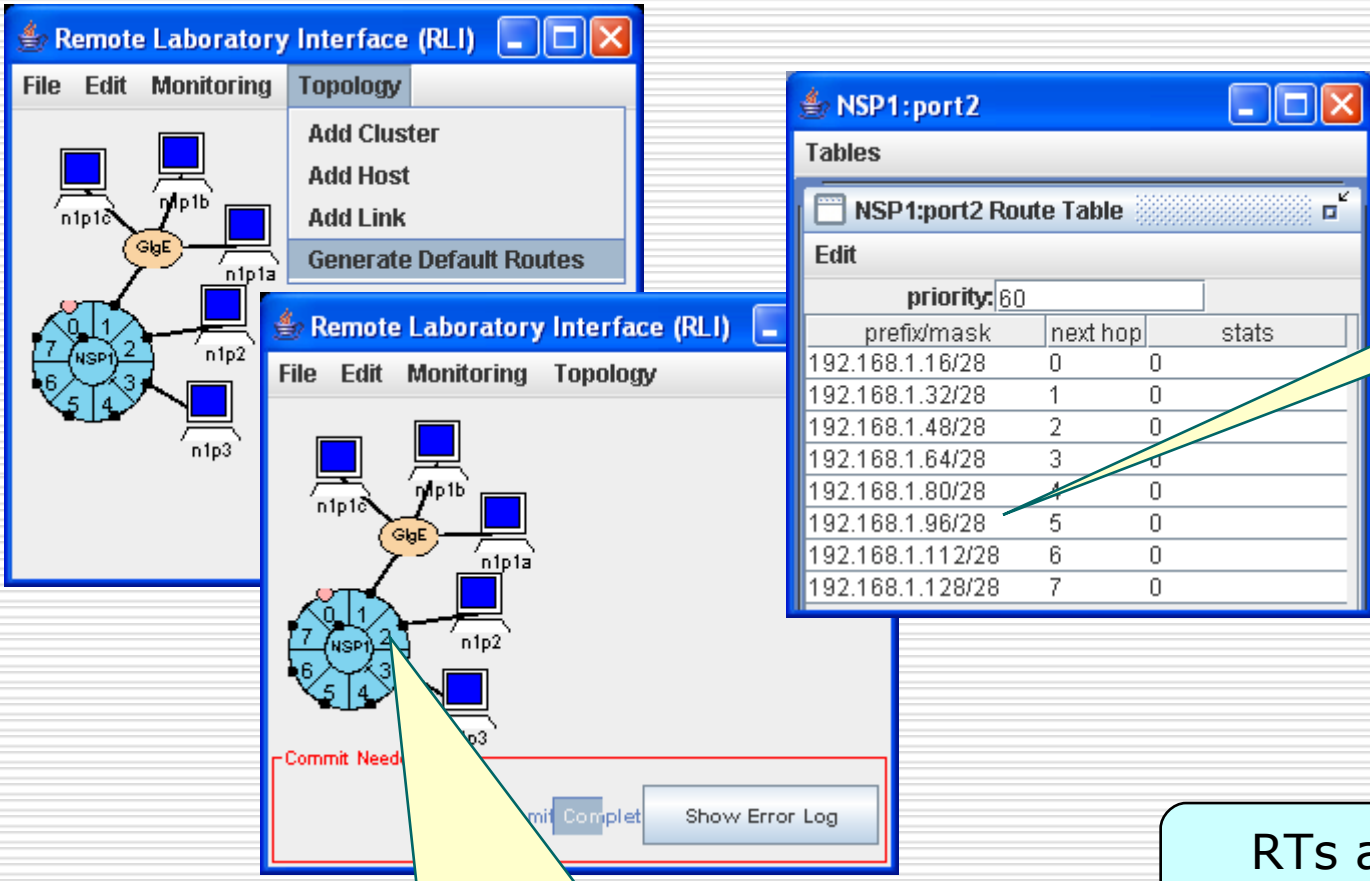
Internal Interface IP Addresses



Control Network: 10.0.0.x

Data Network: 192.168.N.Y
where N = NSP#
Y = 16 x (1 + port#)

Default Route Tables



CIDR network addresses

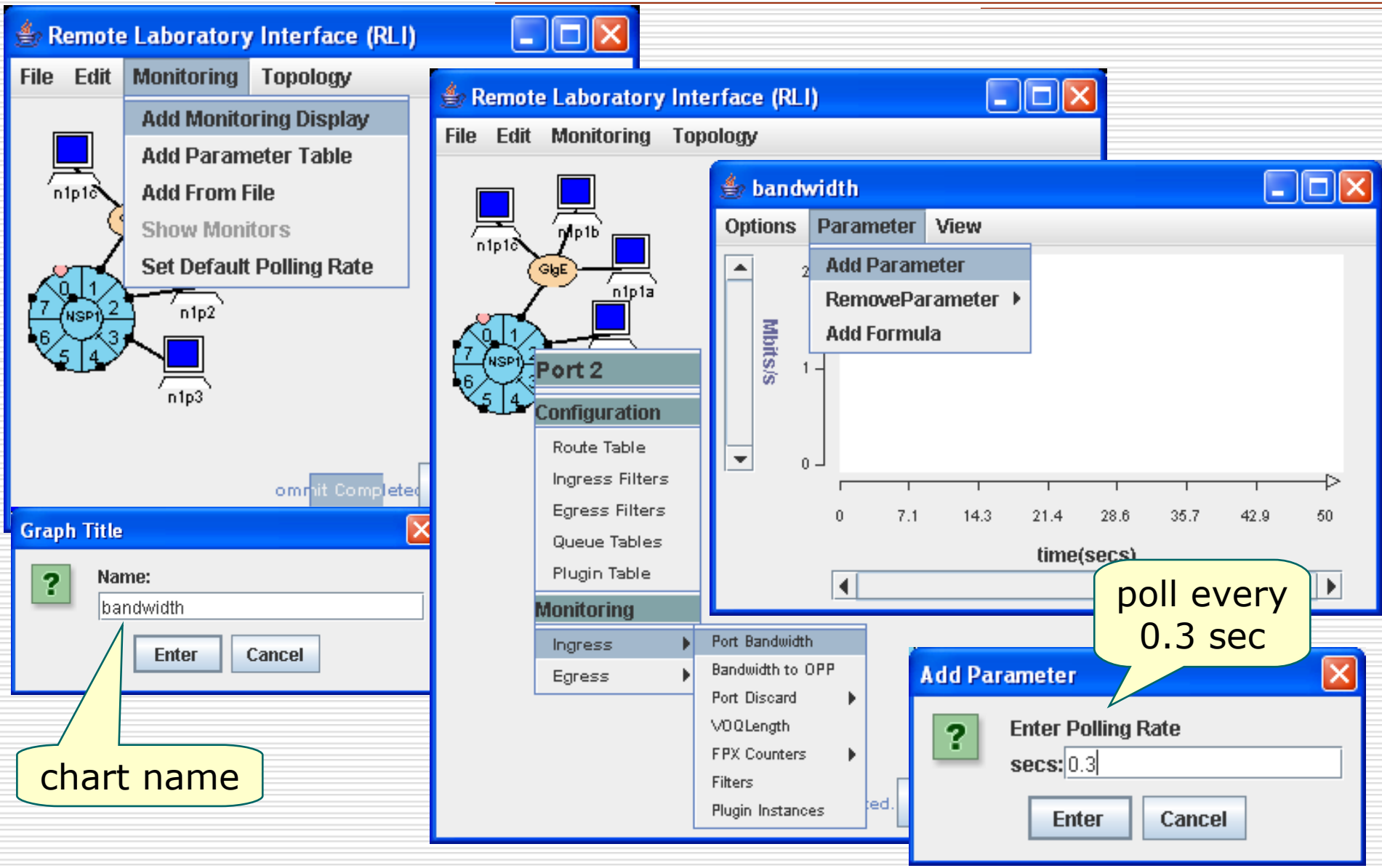
RTs are really Forwarding Tables!

Select
Port 2 => Route Table

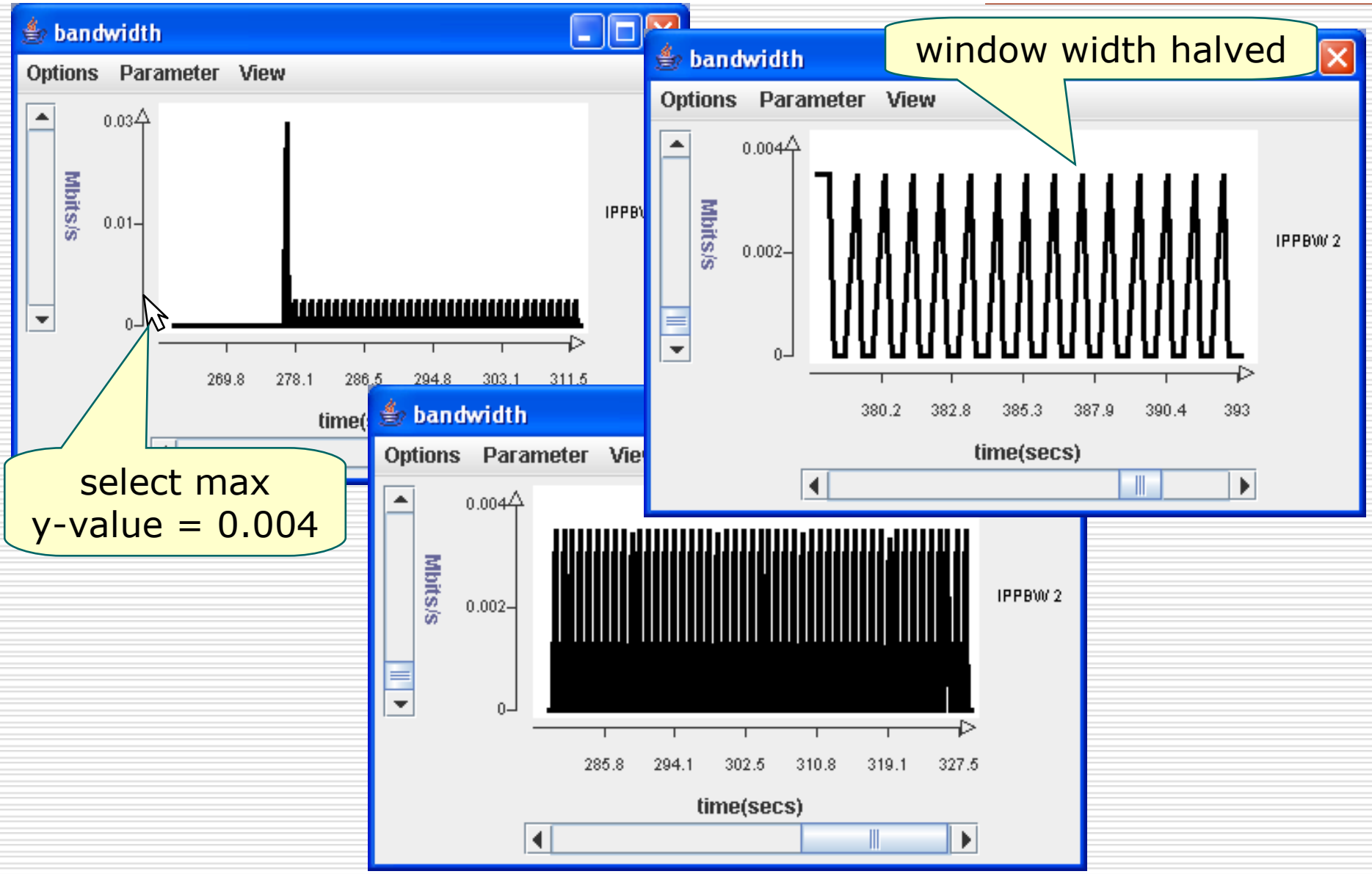
Demo 2

- Simple traffic monitoring
- Add links

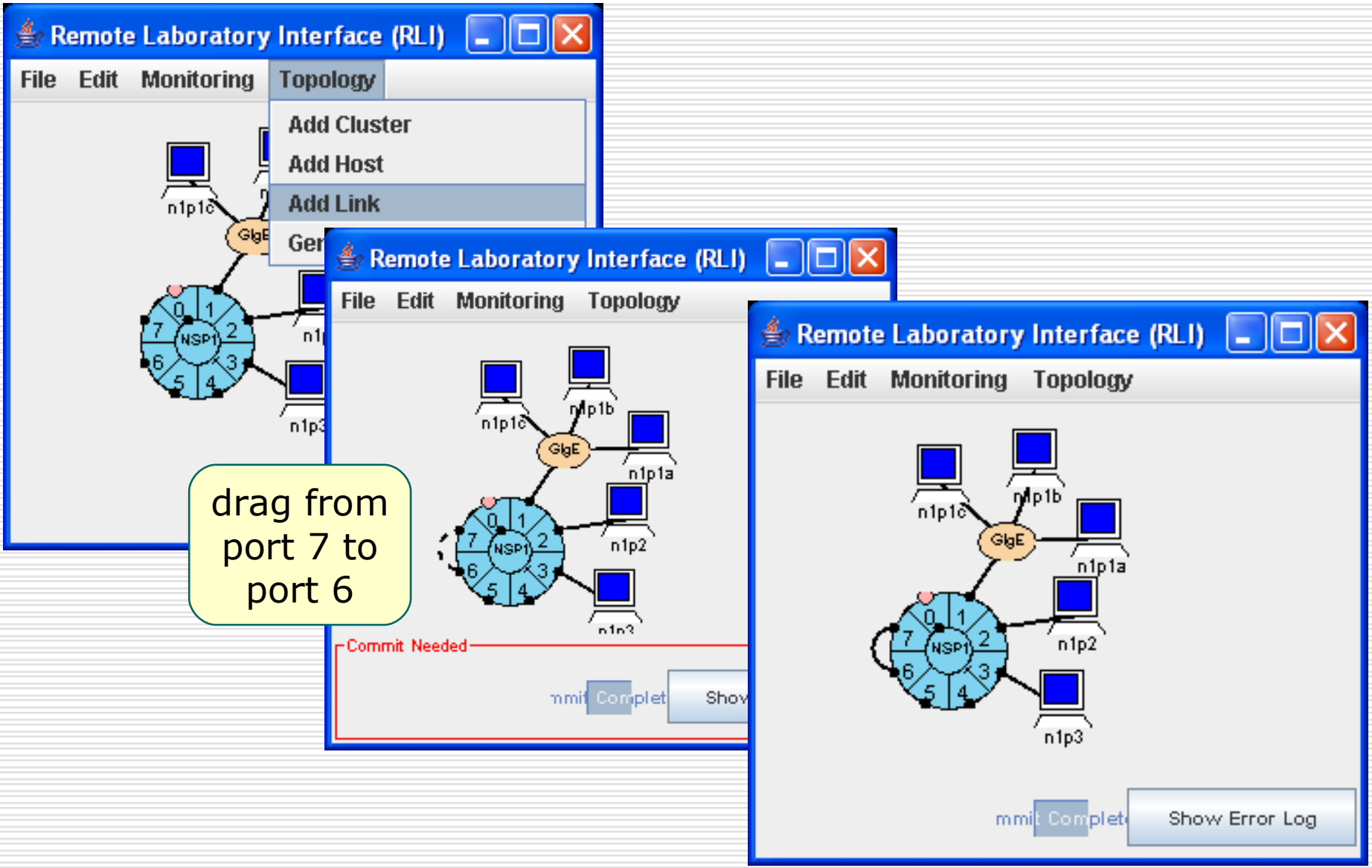
Add Monitoring Display



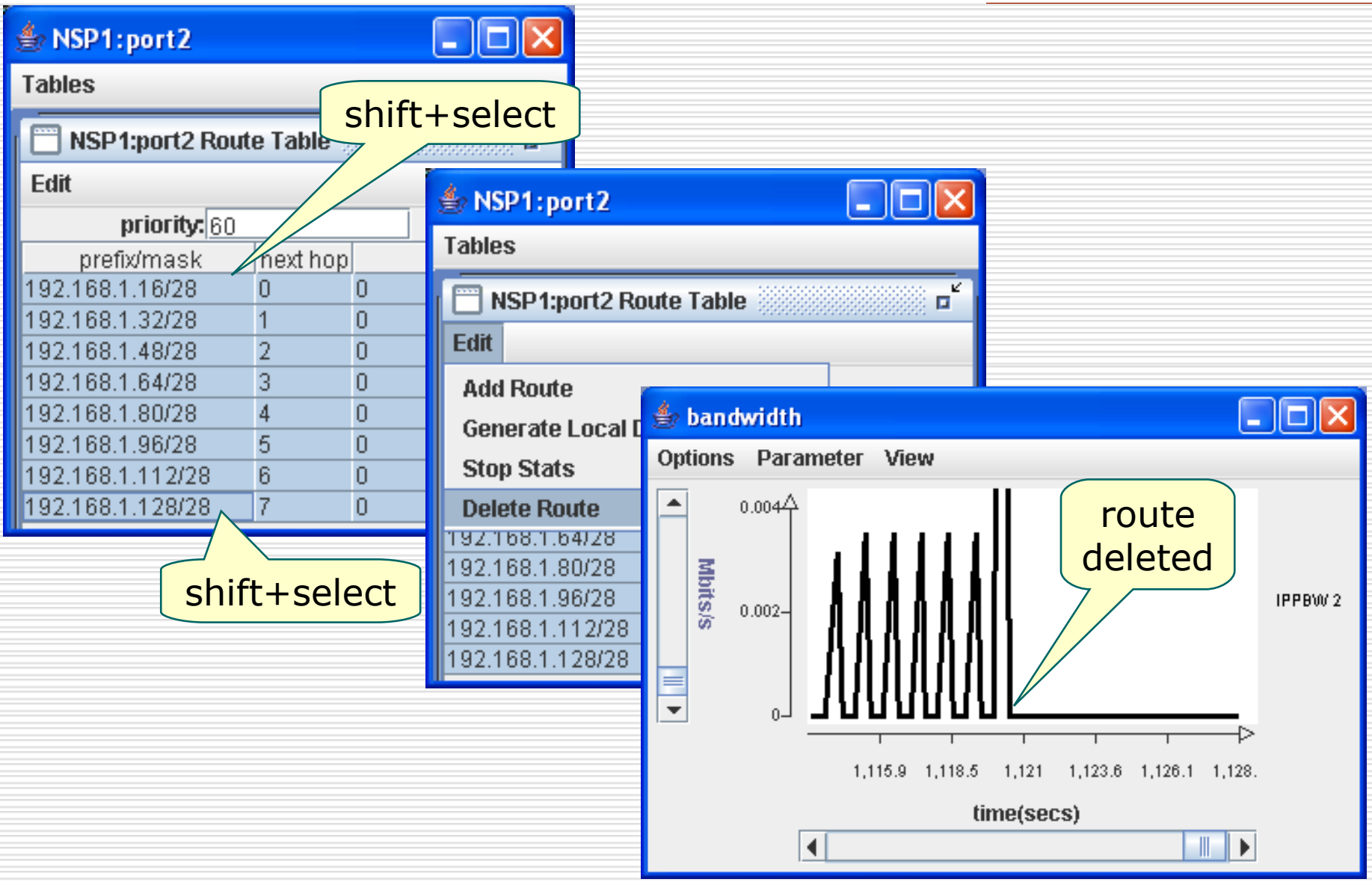
Ping Chart


bandwidth
Options Parameter View
Mbits/s
0.03
0.01
0
IPPBW
269.8 278.1 286.5 294.8 303.1 311.5
time(
select max
y-value = 0.004
bandwidth
Options Parameter View
Mbits/s
0.004
0.002
0
IPPBW 2
380.2 382.8 385.3 387.9 390.4 393
time(secs)
window width halved
bandwidth
Options Parameter Vie
Mbits/s
0.004
0.002
0
IPPBW 2
285.8 294.1 302.5 310.8 319.1 327.5
time(secs)

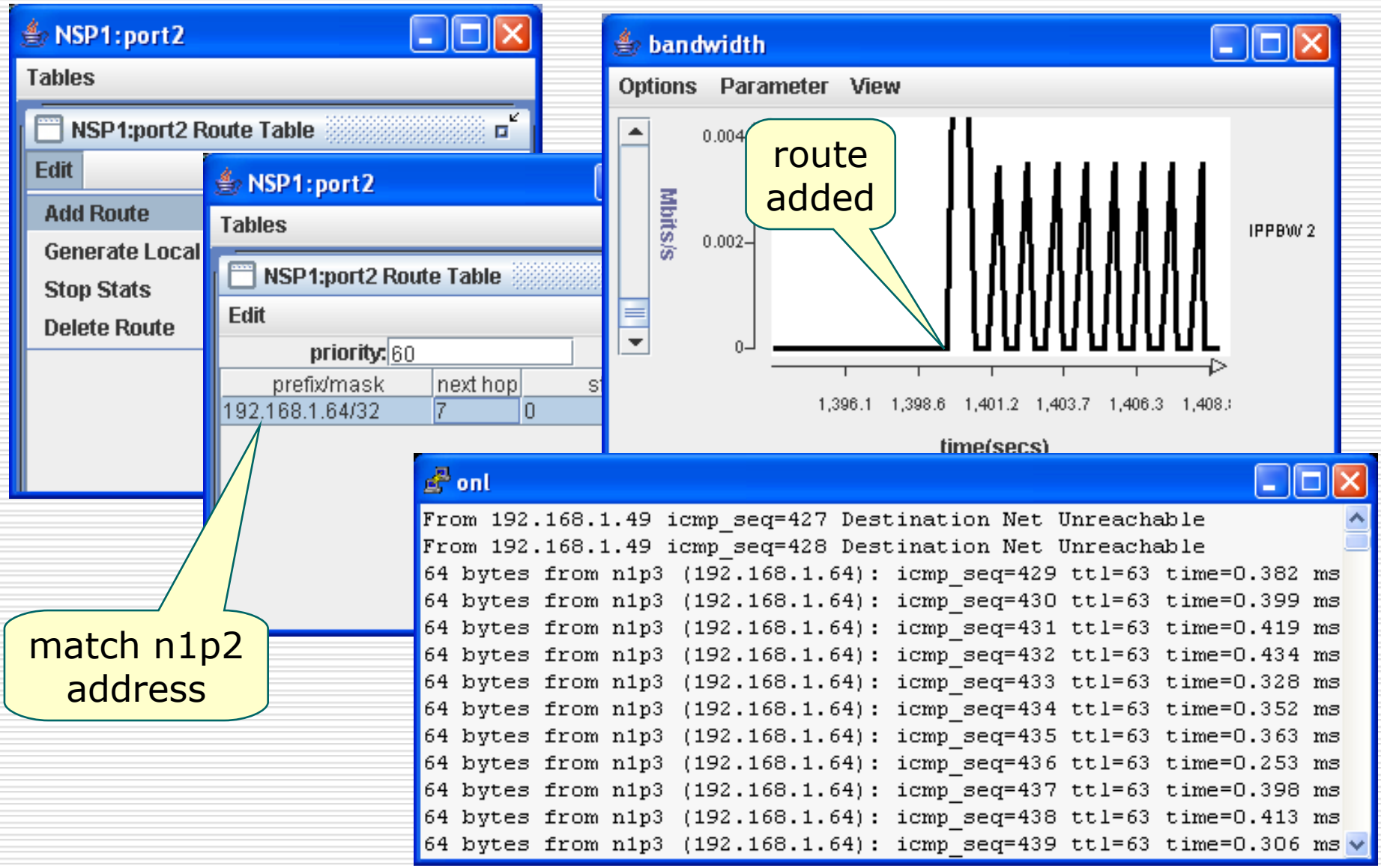
Adding A Link



Delete Route Group



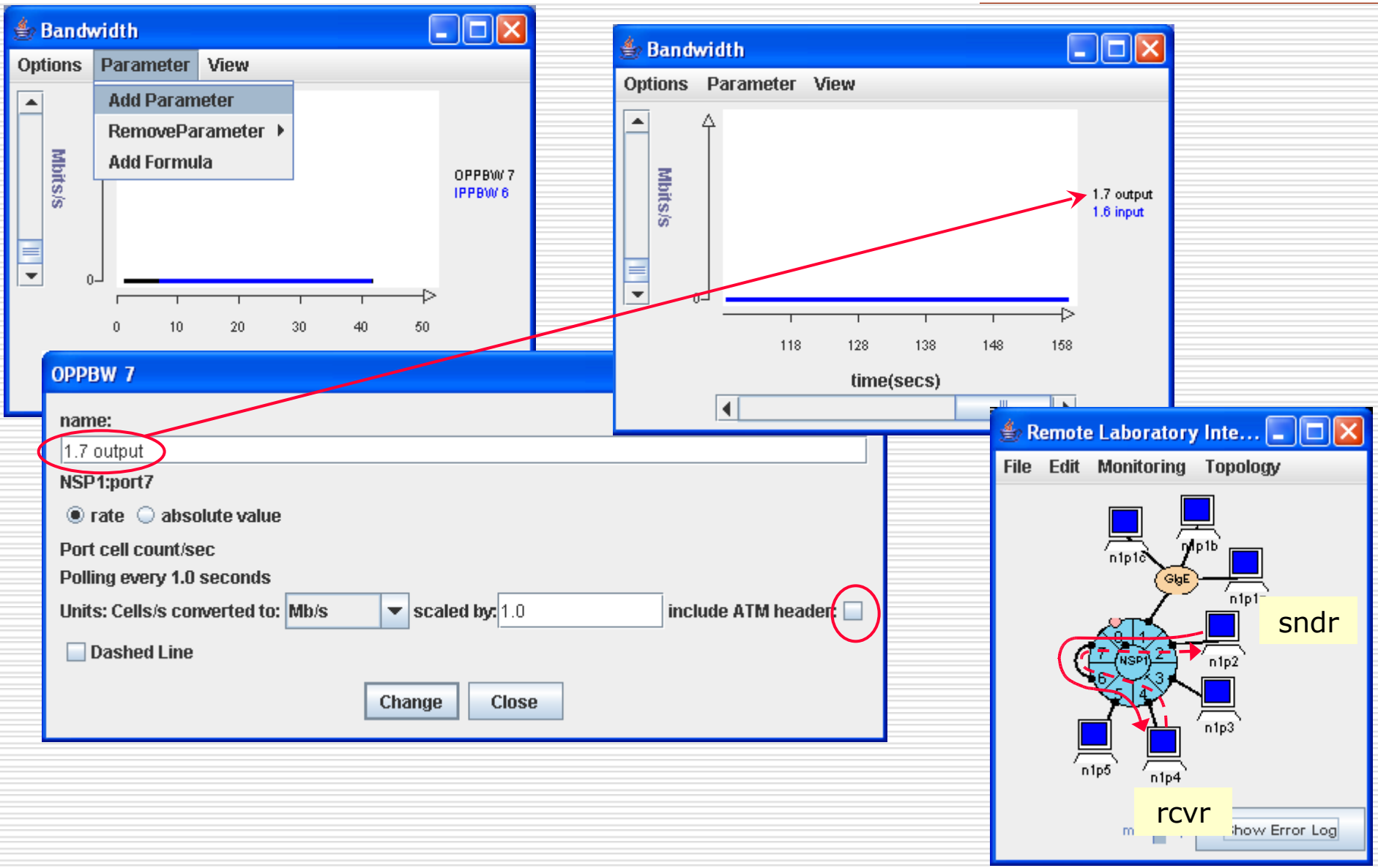
Add Route To Loopback



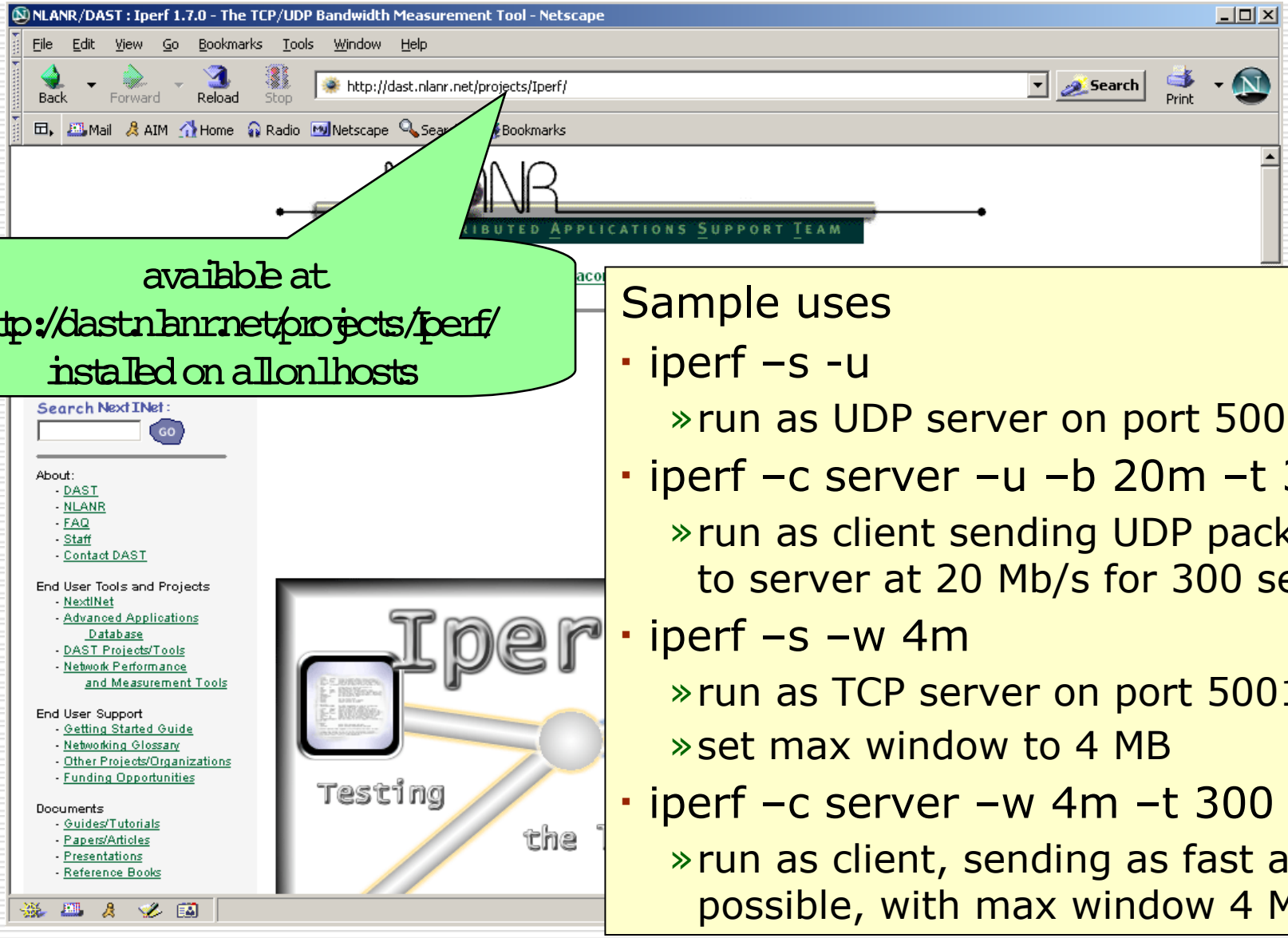
Demo 3

- Traffic monitoring
- iperf traffic generator
- Link capacity

Bandwidth Monitoring



Generating Traffic with Iperf



available at
http://dast.nlanr.net/projects/Iperf/
installed on all onlhosts

Sample uses

- iperf –s -u
  - » run as UDP server on port 5001
- iperf –c server –u –b 20m –t 300
  - » run as client sending UDP packets to server at 20 Mb/s for 300 secs.
- iperf –s –w 4m
  - » run as TCP server on port 5001
  - » set max window to 4 MB
- iperf –c server –w 4m –t 300
  - » run as client, sending as fast as possible, with max window 4 MB

iperf Traffic Generator

```
onl
onl37> iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  108 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.64 port 5001 connected with 192.168.1.48 port 32769
[ ID] Interval       Transfer     Bandwidth       Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec   117 MBytes  98.1 Mbits/sec  0.052 ms 88970/172413 (52%)
```

Start UDP server (receiver)

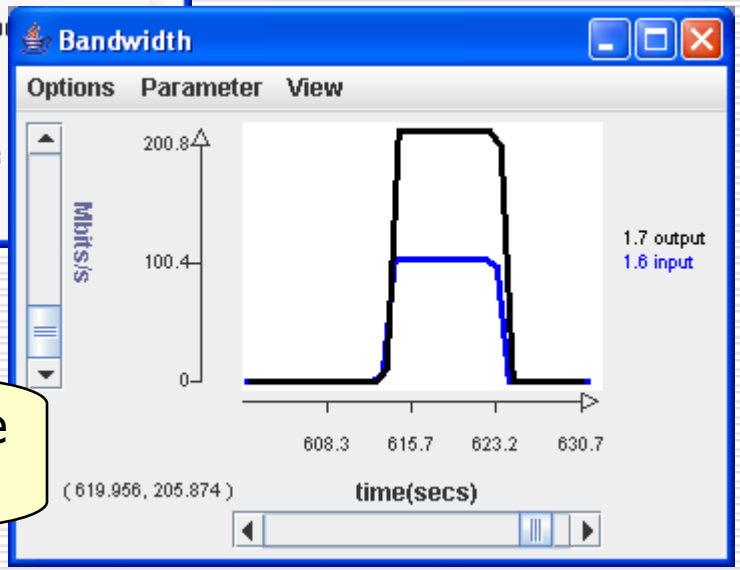
server report

```
onl
onl38> iperf -c n1p3 -u -b 200m -t 10
------------------------------------------------------------
Client connecting to n1p3, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  108 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.48 port 32769 connected with 192.168.1.64 port 50
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   242 MBytes  203 Mbits/sec
[  3] Server Report:
[  3]  0.0-10.0 sec   117 MBy     98.1 Mbits/sec  0.051 ms 88970/172413
[  3] Sent 172413 datagram
onl38>
```

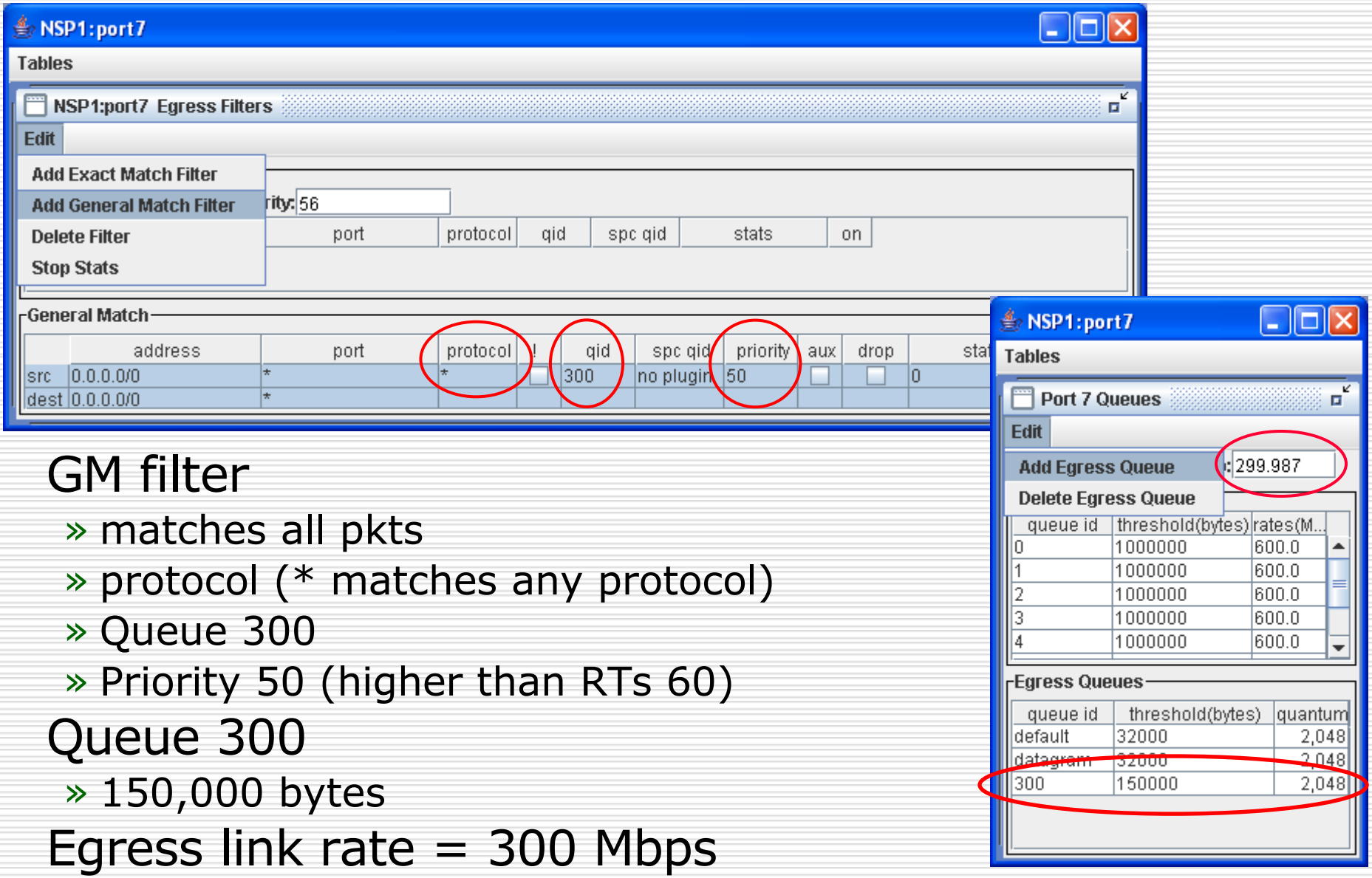
Start client (sender)
200 Mbps bw for 10 sec

client sending rate

server receive rate
(bottleneck)



Adding a General Match Filter



GM filter

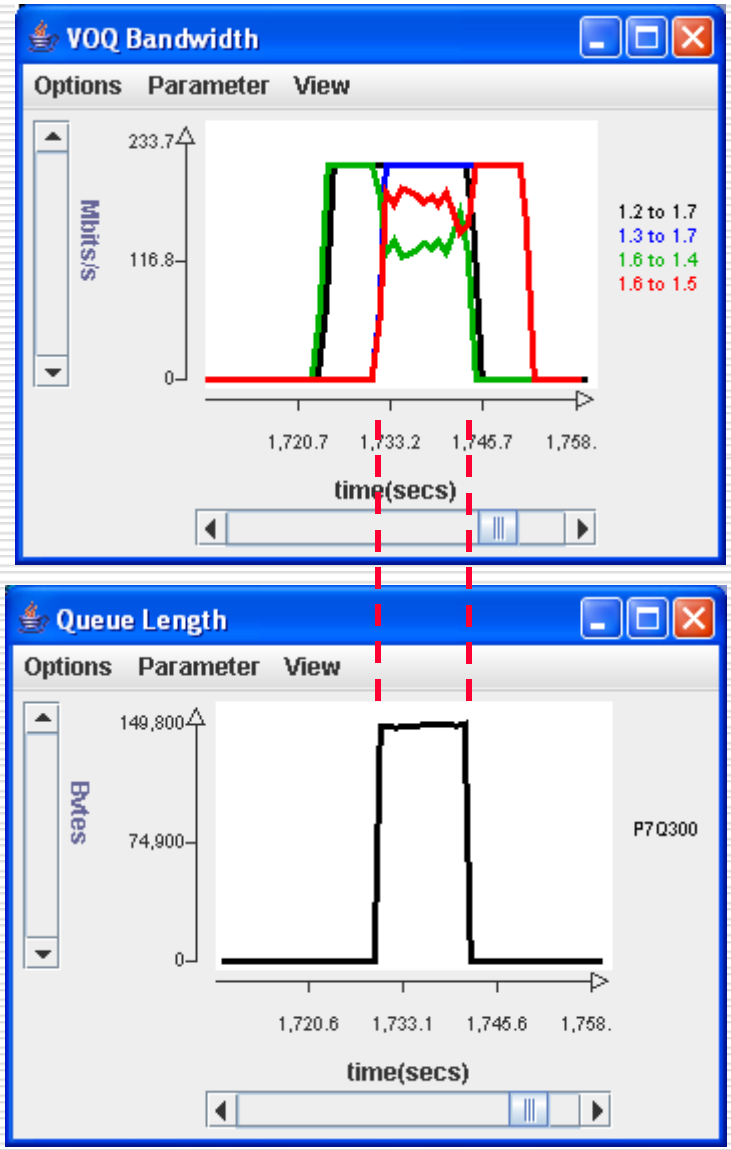
- » matches all pkts
- » protocol (* matches any protocol)
- » Queue 300
- » Priority 50 (higher than RTs 60)

Queue 300

- » 150,000 bytes

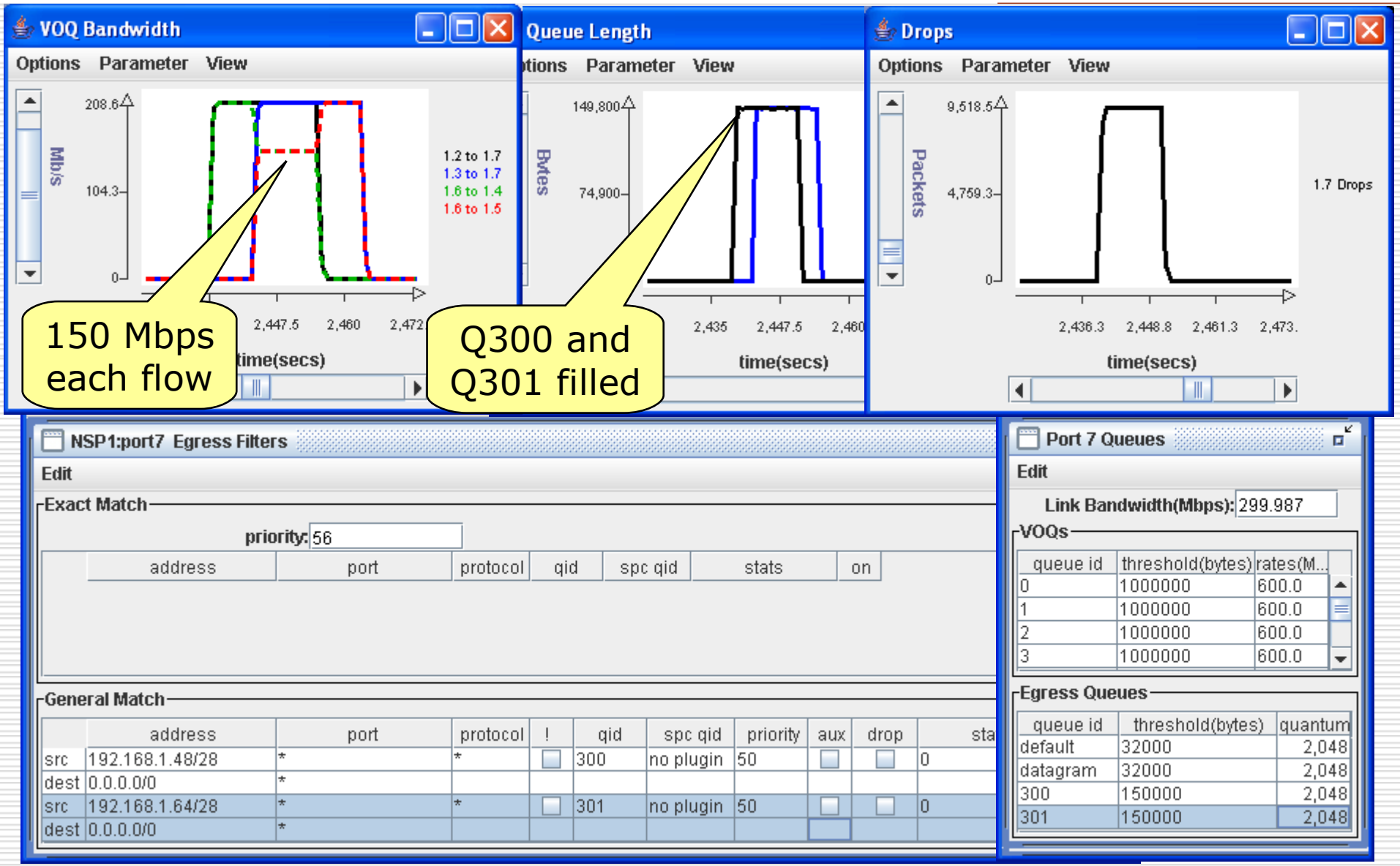
Egress link rate = 300 Mbps

VOQ Bandwidth and Q300 Length



- Both senders send at 200 Mbps
  - » 1.2 to 1.7 and 1.3 to 1.7
- Only n1p2 traffic
  - » 200 Mbps goes to n1p4
- n1p2 and n1p3 traffic
  - » 120 Mbps to n1p4 (1.6-1.4)
  - » 180 Mbps to n1p5 (1.6-1.5)
  - » Queue 300 is full
- Only n1p3 traffic
  - » 200 Mbps goes to n1p5

Separate Queues For Each Flow



Summary

- Real hardware with real-time displays
- Problems?
  - » Web pages have LOTS of information
    - FAQs
    - Examples (used in these demos)
    - Tutorial
  - » Try out a few ideas and see what happens
  - » TAs can help

The End