

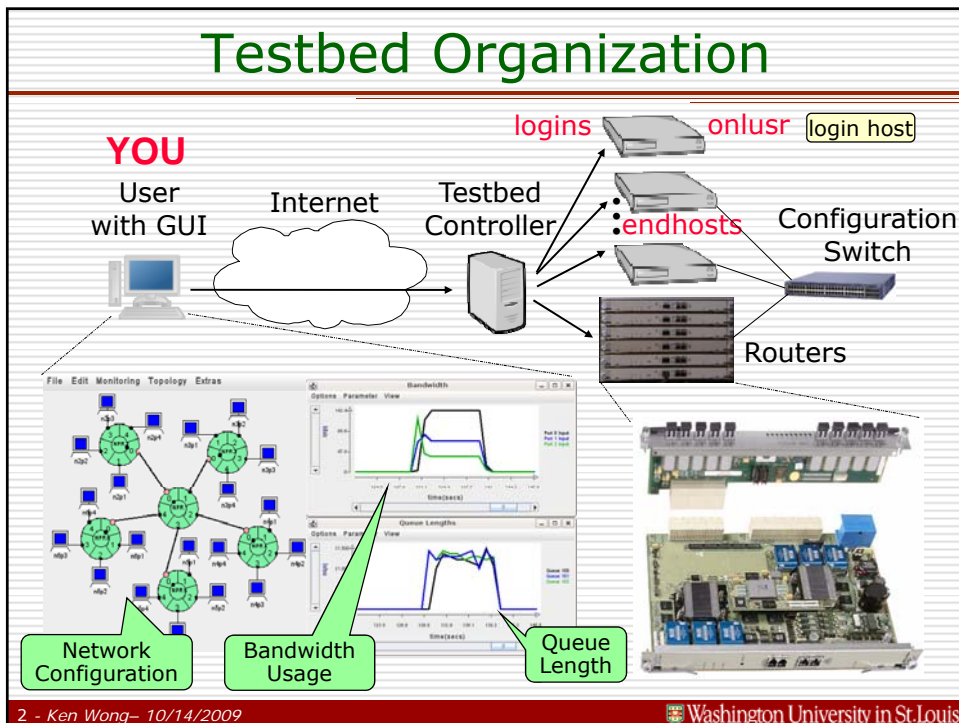
CSE 473S (Fall 2009) The Open Network Lab (Part 1)

Ken Wong
Applied Research Laboratory
Computer Science and Engineering Department

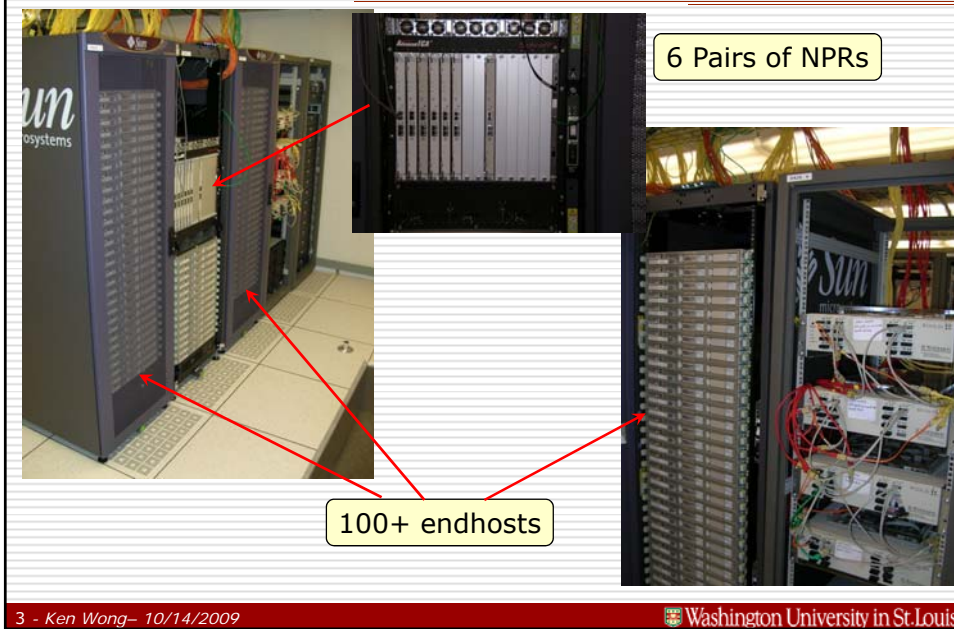
<http://www.arl.wustl.edu/~kenw>
kenw@arl.wustl.edu
<http://www.onl.wustl.edu> (ONL)

National Science Foundation ANI-023826,
CNS-0551651, REL-0632580

 Washington University in St. Louis



Network Processor Routers (NPRs)



Why Build ONL?

- Hands-on experience with real hardware
 - » Make education more concrete → reinforce concepts
 - » Achieve through virtualization
 - Need to protect user from actions of other users
 - Need to mimic propagation delay and link capacity
- Support experiment/observation approach
 - » Extensive monitoring facility
 - » Observe effects of config. changes on network traffic
 - → insights and greater understanding through observations
 - » Easy-to-use Remote Laboratory Interface (RLI)
- Access to advanced router features
 - » Gbps links, filters, packet scheduling
 - » Plugins: Program insertion along packet data path

Get An Account

onl.arl.wustl.edu,
onl.wustl.edu,
www.onl.wustl.edu

tutorial

get an account

Group: onluser
24-hour confirmation
1 password for web & hosts

5 - Ken Wong - 10/14/2009 Washington University in St. Louis

Properties of ONL Accounts

- Same password for Web login and host login
- Host account restrictions
 - » Can only SSH to onl.arl.wustl.edu from remote host
 - Actually end up on onlusr, the ONL user host
 - » **Firewall** blocks all connections from within ONL
 - Can't push from ONL host to remote host
 - Can't pull to ONL host from remote host
 - But can push to ONL host from remote host (e.g., scp)
 - But can pull from ONL host from remote host
 - No email
 - » Can only access onlusr host and hosts assigned to your experiment
- Password-free SSH between ONL hosts
 - » See "NPR Tutorial → Summary Information → Password-Free SSH"

Getting Started

- See sidebar at www.onl.wustl.edu
- Follow instructions in “Getting Started” link
- Read NPR Tutorial
 - » Packet Processing
 - » The Remote Laboratory Interface
 - » Filters, Queues and Bandwidth
- Other useful tutorial sections
 - » Examples → The Remote Laboratory Interface
 - » Examples → Filters, Queues and Bandwidth

SSH Tunnel Configuration

- *Build before each experimental session*
 - » Allows your RLI to communicate with ONL daemon
 - » Needed to make reservation or run experiment
- Cookbook for 3 approaches
 - » *Unix or Unix-like* (e.g., cygwin)
 - `ssh -L 7070:onlsrv:7070 onl.arl.wustl.edu`
 - I use this (defined as an alias) from Linux
 - » *MS Windows PuTTY* (GUI-based)
 - I use this from my Microsoft XP laptop
 - And also the SSH agent *pageant* (optional)
 - See Tutorial for URL (free software)
 - » *Microsoft Windows SSH client tool from SSH Corp*
- Follow instructions on ONL web pages
 - » *Don't agonize over it → See TA after 2-3 tries*
 - » If problems, send email to your grader/consultant

PuTTY SSH Tunnel Configuration

external host name = onl.arl.wustl.edu

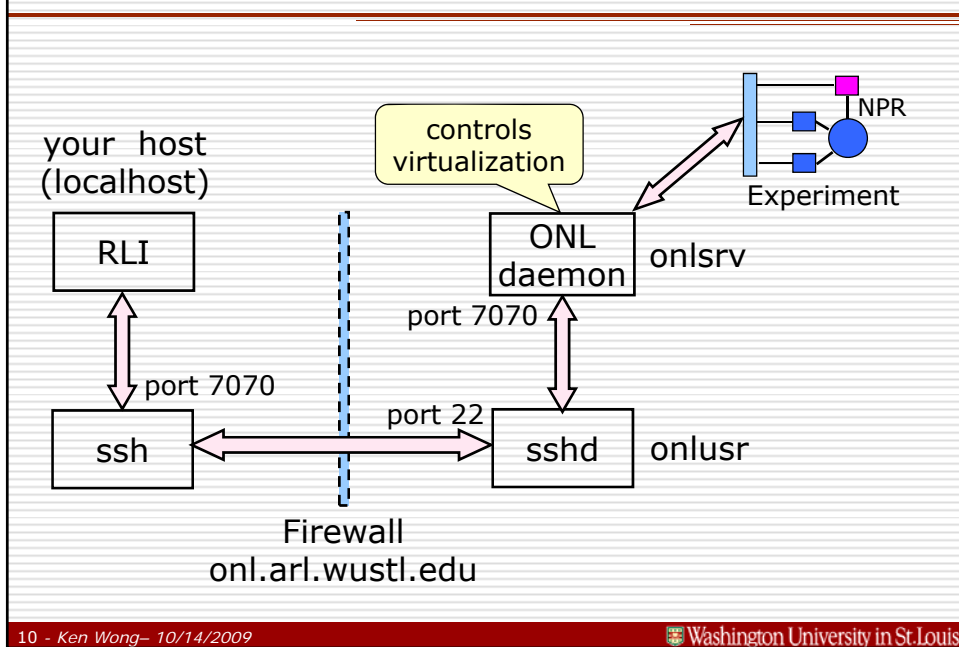
session name = rli

local port 7070
remote host onlsvr
remote port 7070

9 - Ken Wong - 10/14/2009

Washington University in St. Louis

Under the SSH Tunnel Hood



Configuring a Virtual IP Network

■ Topology

- » NPRs, hosts, links
- » Used for *resource reservation*

■ Route Tables

- » per port
- » *Initially empty at all ports*
- » *default RT*: 1 port or all ports
- » IP pkt forwarding
 - get pkt 1 hop closer to destination
- » uses Longest Prefix Match

■ Filter Tables

- » per port
- » forward to port/queue
- » forward to plugin, port/queue

■ Queue Tables

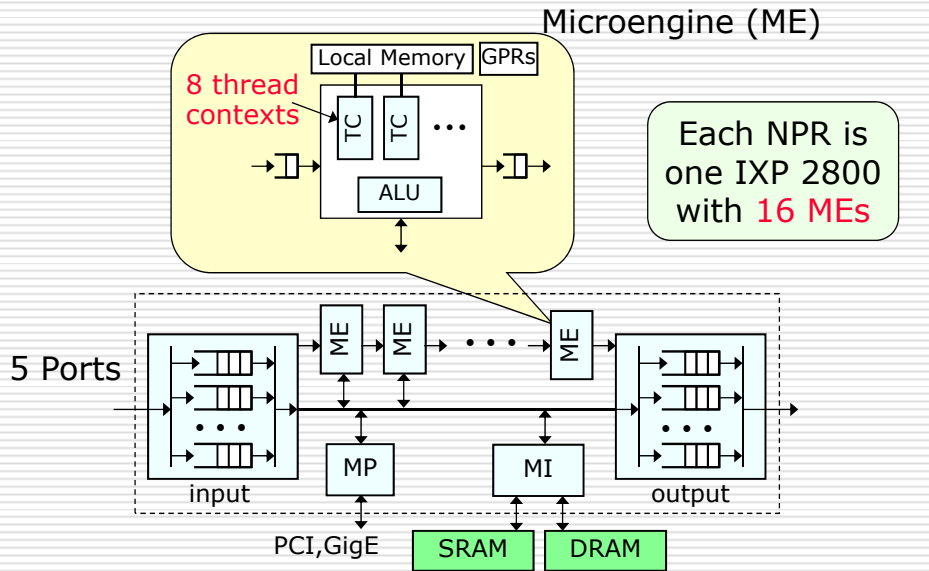
- » per port
- » port rate
- » datagram queues
 - QIDs 0-63
- » reserved queues
 - QIDs 64-8,191
- » queue parameters
 - *threshold*: queue size
 - *quantum*: scheduling weight

■ Plugin Tables

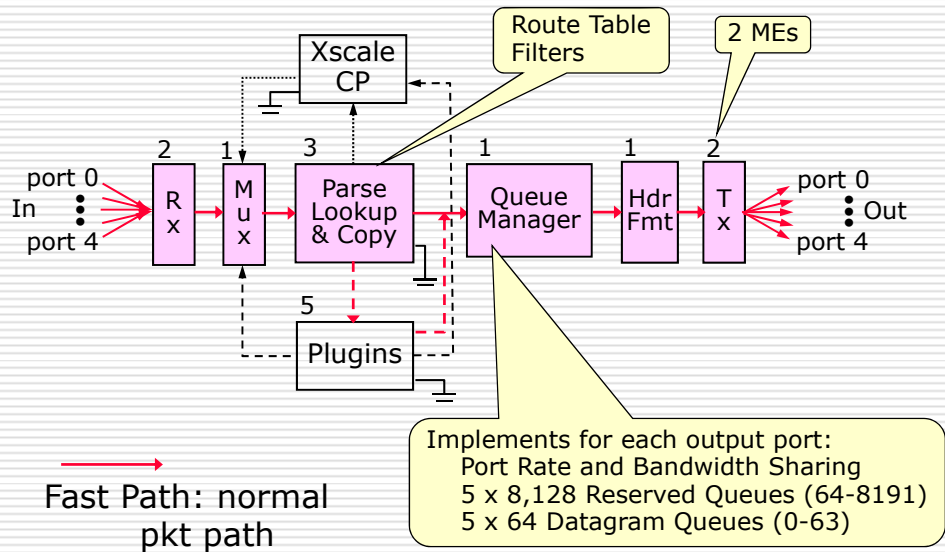
- » per NPR
- » standard plugins
 - e.g., delay
- » user-defined plugins

The NPR Data Path

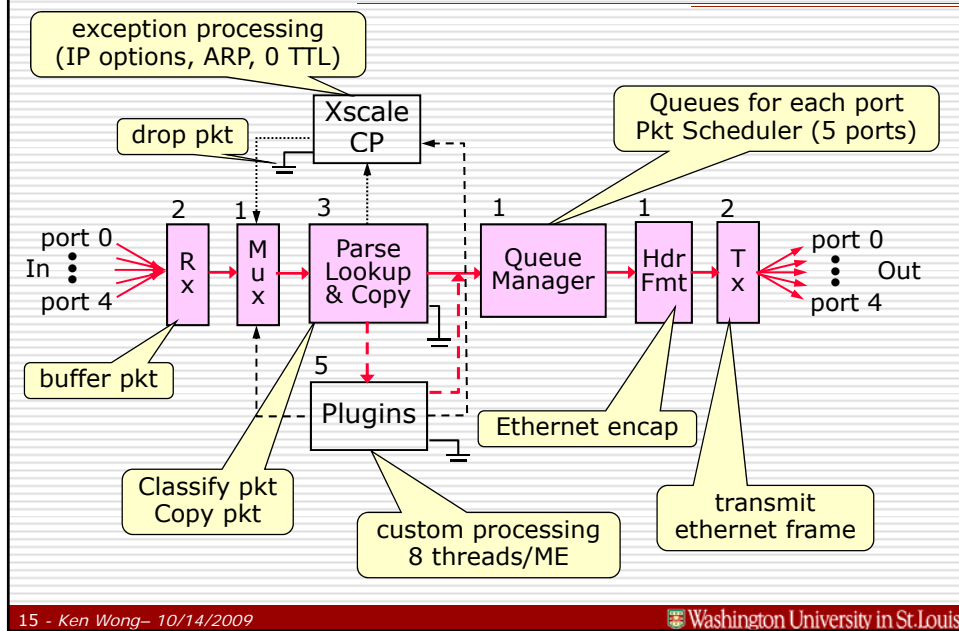
The Network Processor Router (NPR)



NPR Packet Processing (1)



NPR Packet Processing (2)



NPR Packet Processing (3)

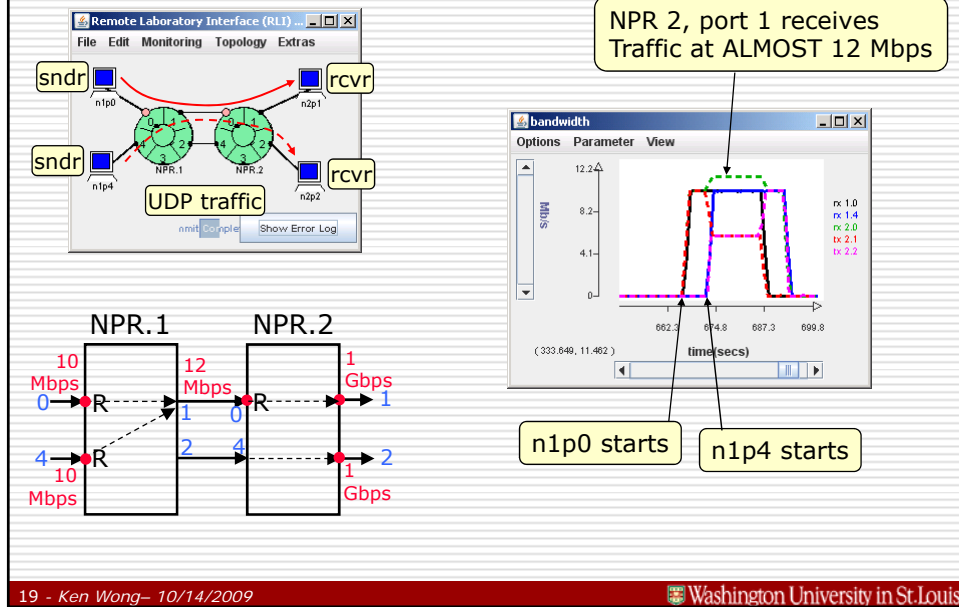
- Rx: Receive Pkt
 - » Put pkt in DRAM; Send **meta-packet**
- Mux: Multiplex traffic (inputs, CP, plugins)
- PLC: Parse, Lookup and Copy
 - » Implements Route Table and Filter Table lookup
 - » Uses TCAM (Ternary Content-Addressable Memory)
- QM: Queue Manager
 - » Pkt scheduler for each of 5 output ports
 - Implements *port rate concept* using *token bucket*
 - Implements *bandwidth sharing concept* using *Weighted Deficit Round Robin* (WDRR) algorithm
- Hdr Format
 - » Create ethernet frame
- Tx: Transmit Ethernet Frame

Experiment 1

Topics

- 2 NPRs, 4 hosts, 2 links between NPRs
- Experiment files
 - » 473-1-f09.exp
 - » 473-1-filters-f09.exp (filters at ports 1.0 and 1.4)
- Unix commands
 - » /sbin/ifconfig network interface properties
 - » /bin/netstat network interface statistics
 - » /bin/ping ICMP echo/reply
 - » /usr/local/bin/iperf UDP/TCP traffic generator
- Route tables and port rate (capacity)

Experiment 1 (Final Version)



Steps in Running an Experiment

- Define *maximal* network resources using RLI
 - » Required: Add NPR routers, Hosts, Links
- Save configuration
 - » File → Save as
- *Build SSH tunnel to onl.wustl.edu*
- Make a reservation
 - » File → Make reservation
 - » Can be an *advanced reservation*
- When reservation time arrives:
 - » *Build SSH tunnel to onl.wustl.edu*
 - » Run RLI and open configuration file
 - » Ask for network to be built: File → Commit
 - » Define more network parameters (Commit) and monitoring
 - » Wait for commit to finish
- Log into onl.wustl.edu using ssh
 - » Prompt will show you have logged into *onlusr*
 - » You can only access your ONL nodes from onlusr host
 - » Use Linux commands to run traffic generators

RLI Menus

- Menu File → ...
 - » *Open, Commit* (binds virtual network to actual network components)
- Port → Route Table → Edit
 - » Add/Delete Route, Generate Local Default Routes
- Port → Queue Table
 - » Port Rate, Edit → Add/Delete Queue, Threshold, Quantum
- Port → Filter Table → Edit
 - » Add/Edit/Delete Filter
- *Menu Monitoring → Add Monitoring Display*

Send ping pkts from n1p0 to n2p1

The screenshot displays the RLI interface with three main components:

- Topology View:** Shows a network diagram with nodes n1p0, n2p1, n1p4, and n2p2. Two network processors, NPR.1 and NPR.2, are connected. A 'ping traffic' label is placed between NPR.1 and NPR.2.
- Bandwidth Monitor:** A graph showing a sharp spike in bandwidth usage at approximately 90 seconds, corresponding to the ping test.
- Terminal Window:** Shows the execution of a ping command from node on1060 to n2p1. The output includes individual ping results and summary statistics.

Terminal Output:

```

onlusr> ssh on1060
Last login: Wed Aug 5 13:54:23 2009 from onlusr.arl.wustl.edu
on1060> ping -c 3 n2p1
PING n2p1a (192.168.2.32): 56(84) bytes of data:
64 bytes from n2p1a (192.168.2.32): icmp_seq=1 ttl=62 time=164 ms
64 bytes from n2p1a (192.168.2.32): icmp_seq=2 ttl=62 time=0.077 ms
64 bytes from n2p1a (192.168.2.32): icmp_seq=3 ttl=62 time=0.076 ms

--- n2p1a ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.076/54.778/164.181/77.359 ms
on1060>
    
```

Annotations in the image include:

- A yellow box labeled "send 3 pkts to n2p1" pointing to the terminal command.
- A yellow box labeled "RTT" pointing to the ping statistics line.
- A yellow box labeled "stats" pointing to the terminal output.

Send UDP pkts from n1p0 to n2p1

The screenshot shows a network lab environment with two hosts, n1p0 and n2p1, connected via a network. The sender (n1p0) is running a script to send 1470 UDP datagrams to n2p1. The receiver (n2p1) is running a script to receive these datagrams. A bandwidth graph shows the transfer rate peaking at 20.4 Mbps. A network topology diagram shows the connection between the two hosts.

```

onlusr> ssh on1060
Last login: Mon Oct 12 12:08:57 2009 from onlusr.arl.wustl.edu
on1060> /usr/local/bin/iperf -c n2p1 -u -w 1m -b 20m -t 20

Client connecting to n2p1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 2.00 MByte (WARNING: requested 1.00 MByte)

[ 3] local 192.168.1.16 port 32769 connected with 192.168.2.32 port 5001
[ 3] 0.0-20.0 sec 47.7 MBytes 20.0 Mbits/sec
[ 3] Sent 34015 datagrams
[ 3] Server Report:
[ 3] 0.0-20.0 sec 27.2 MBytes 11.4 Mbits/sec 0.221 ms 14594/34013 (43%)
[ 3] 0.0-20.0 sec 1 datagrams received out-of-order
on1060>
    
```

server got 11.4 Mbps

sent at 20 Mbps

```

onlusr> ssh on1067
Last login: Fri Oct 9 15:25:02 2009 from onlusr.arl.wustl.edu
on1067> /usr/local/bin/iperf -s -u -w 1m

Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 2.00 MByte (WARNING: requested 1.00 MByte)

[ 3] local 192.168.2.32 port 5001 connected with 192.168.1.16 port 32769
[ 3] 0.0-20.0 sec 27.2 MBytes 11.4 Mbits/sec 0.221 ms 14594/34013 (43%)
[ 3] 0.0-20.0 sec 1 datagrams received out-of-order
    
```

Options	Parameter	View
rx	1.0	
rx	1.4	
rx	2.0	
tx	2.1	
tx	2.2	

23 - Ken Wong - 10/14/2009 Washington University in St. Louis

ifconfig and netstat commands

The screenshot shows a terminal window where the user configures two network interfaces, eth0 and eth1, and then runs netstat to view the kernel interface table.

```

onlusr> ssh on1060
Last login: Mon Oct 12 12:11:33 2009 from onlusr.arl.wustl.edu
on1060> /sbin/ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:EO:81:5E:9C:B7
          inet addr:10.0.0.60  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3448 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2844 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:578093 (564.5 KiB)  TX bytes:290494 (283.6 KiB)
          Interrupt:20

on1060> /sbin/ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 00:EO:81:5E:9C:B6
          inet addr:192.168.1.16  Bcast:192.168.1.16  Mask:255.255.255.255
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34024 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1926 (1.8 KiB)  TX bytes:5156882 (49.1 MiB)
          Interrupt:16 Base address:0x4000

on1060>
    
```

eth0 control interface 10.0.0.90

eth1 data interface 192.168.1.16

```

on1060> /bin/netstat -i
Kernel Interface table
Iface      MTU Met  RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500  0    3540   0     0     0    2904   0     0     0  BMRU
eth1       1500  0     6     0     0     0   34024   0     0     0  BMRU
lo         16436  0     90    0     0     0     90    0     0     0  LRU
    
```

/bin/netstat

24 - Ken Wong - 10/14/2009 Washington University in St. Louis

Route Tables

all RTs start empty

add/delete entries

affects every port, every NPR

to NPR 2

output port

25 - Ken Wong - 10/14/2009

Washington University in St. Louis

Queue Tables and Port Rate

Default port rate = 1 Gbps

Requested 12 Mbps

» actual max port rate after commit = 11.611 Mbps

- 683 Kbps granularity
- 11.611 Mbps = 17 x 0.683 Mbps

26 - Ken Wong - 10/14/2009

Washington University in St. Louis

Bandwidth Monitoring (1)

Remote Laboratory Interface (RLI) ...

File Edit Monitoring Topology Extras

Add Monitoring Display
Add Parameter Table
Show Monitors
Set Default Polling Rate

Graph Title
Name: bandwidth

Add Parameter
secs: 1
 rate

bandwidth
Options Parameter View
Add Parameter
Remove parameter
Add Formula

Index/s
time(secs)

27 - Ken Wong - 10/14/2009

Washington University in St. Louis

Bandwidth Monitoring (2)

bandwidth
Options Parameter View
Show Values
Reset View
Change Y-axis label
Set Log Scale
Zoom In From Selection

bandwidth: change y-axis label
Units: Mb/s
optional label:
default label is unit type.

change Y-axis label

select legend label to modify chart properties

rx 1.0
name: rx 1.0
NPR.1:port0
Port 0
read rx byte count/sec
 rate absolute value
Polling every 1.0 seconds
Units: bytes/s converted to: Mb/s scaled by: 1.0

28 - Ken Wong - 10/14/2009

Washington University in St. Louis

Experiment 1 Shell Script

See `~kenw/bin/run-473`

Puts `$n1p0`, etc into Unix environment

start 2 UDP receivers

start 1st UDP sender

wait 6 seconds

start 2nd UDP sender

```
onl100.arl.wustl.edu - PuTTY
onlusr> ls -l bin/run-473
-rwxr--r-- 1 kenw onluser 405 2009-10-11 22:16 bin/run-473
onlusr> cat bin/run-473
#!/bin/sh
#
# Purpose:      run cse473s experiment 1
#
source /users/onl/.topology # get defs of external interfaces

# start 2 udp receivers
for host in $n2p1 $n2p2; do
  ssh $host /usr/local/bin/iperf -s -u -w 1m &
done
sleep 1

# start 2 udp senders staggered by 6 sec
ssh $n1p0 /usr/local/bin/iperf -c n2p1 -u -w 1m -b 10m -t 20 &
sleep 6
ssh $n1p4 /usr/local/bin/iperf -c n2p2 -u -w 1m -b 10m -t 20 &
onlusr>
```

10 Mbps
20 sec
run as client; send to n2p2
UDP
1 MB buffer

Experiment 1 Shell Script

See `~kenw/bin/run-473`

Puts `$n1p0`, etc into Unix environment

start 2 UDP receivers

start 1st UDP sender

wait 6 seconds

start 2nd UDP sender

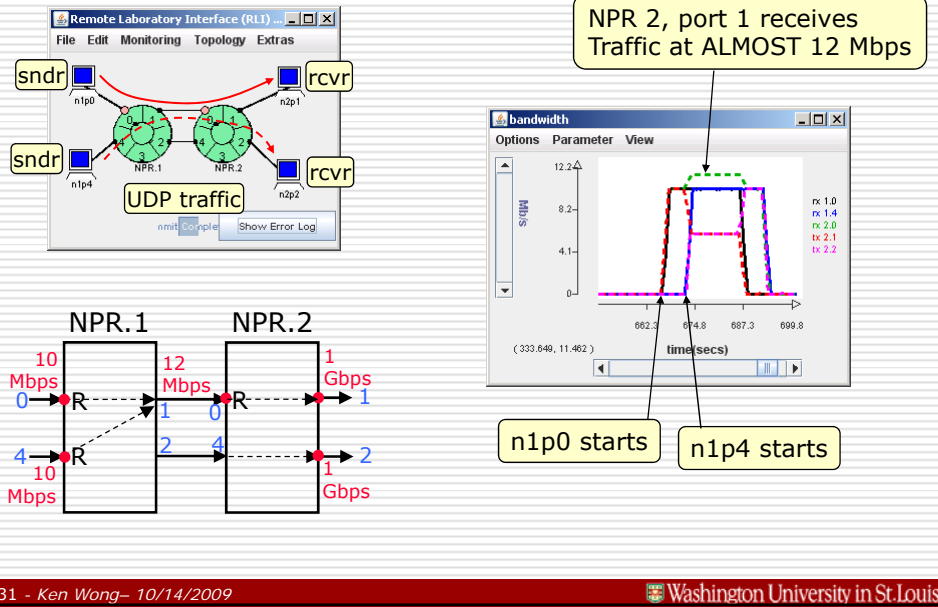
```
onl100.arl.wustl.edu - PuTTY
onlusr> ls -l bin/run-473
-rwxr--r-- 1 kenw onluser 405 2009-10-11 22:16 bin/run-473
onlusr> cat bin/run-473
#!/bin/sh
#
# Purpose:      run cse473s experiment 1
#
source /users/onl/.topology # get defs of external interfaces

# start 2 udp receivers
for host in $n2p1 $n2p2; do
  ssh $host /usr/local/bin/iperf -s -u -w 1m &
done
sleep 1

# start 2 udp senders staggered by 6 sec
ssh $n1p0 /usr/local/bin/iperf -c n2p1 -u -w 1m -b 10m -t 20 &
sleep 6
ssh $n1p4 /usr/local/bin/iperf -c n2p2 -u -w 1m -b 10m -t 20 &
onlusr>
```

run as client; send to n2p2
UDP
1 MB buffer

Experiment 1 (Final Version)



Experiment 2

Using Reserved Queue 64

add reserved queue 64 (600,000 bytes)

queue id	threshold(bytes)	quantum
default	32768	1500
64	600000	1500

Add Filter Configuration:

- source address/mask: 0.0.0.0/0
- destination address/mask: 0.0.0.0/0
- protocol: *
- output ports: 1
- qid: 64
- priority(0-63): 50

Commit Needed

Monitoring Queue Length

Graph Title: Queue Length

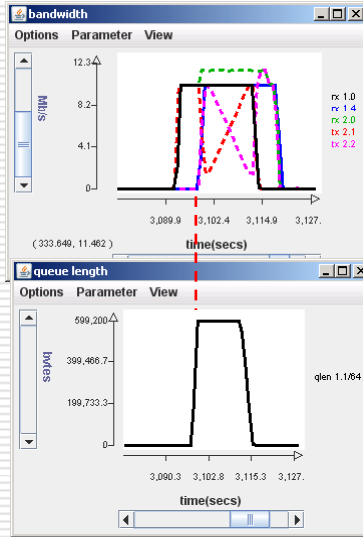
Add Parameter Configuration:

- queue_id: 64
- Enter Polling Rate: secs: 1

Queue Length Graph:

The graph shows a plot of Queue Length (Y-axis, 0 to 2) versus time (X-axis, 3.793.4 to 3.835 seconds). The graph is currently empty.

Experiment 2 Displays



Delaying Packets

Delaying Packets From n2p1

The image shows a network topology in the Remote Laboratory Interface (RLI) with two routers, NPR.1 and NPR.2, connected to hosts n2p1 and n2p4. A yellow box notes: "standard plugins ~onl/npr/plugins/". A menu for "Router NPR.2 Plugins" is open, showing options like emumk, damage, null2, nstats, ipHdr, delay, debug, and count. A diagram below shows traffic flow from n2p4 through NPR.2 (ports 1, 2, 3, 4) and NPR.1 (ports 1, 2, 3, 4) to rcvr. A yellow box notes: "Add a delay plugin D" and "Add a filter F". A "Router NPR.2 P..." window shows "plugin micro e..." and "delay 0". A yellow box notes: "5 possible plugin MEs numbered 0-4" and "microengine 0".

Add The Filter

The screenshot shows the Remote Laboratory Interface (RLI) configuration for a network. On the left, a topology diagram shows two nodes, NPR.2 and NPR.1, connected. NPR.2 has IP Base Address: 192.168.2.0 and CP host: nprop12.arl.wustl.edu. Below the topology is a diagram of the filter configuration: NPR.2 has ports 1, 2, 3, and 4. Port 1 is connected to a filter 'F' with a red 'D' (Drop) and a green 'Q300' (Queue). Port 2 is connected to a filter 'F' with a red 'D'. Port 3 is connected to a filter 'F' with a red 'D'. Port 4 is connected to a filter 'F' with a red 'D'. NPR.1 has ports 1, 2, 3, and 4. Port 1 is connected to a filter 'F' with a red 'D'. Port 2 is connected to a filter 'F' with a red 'D'. Port 3 is connected to a filter 'F' with a red 'D'. Port 4 is connected to a filter 'F' with a red 'D'.

The 'Add Filter' dialog box is open, showing the following configuration:

- source address/mask: 0.0.0.0/0
- destination address/mask: 0.0.0.0/0
- protocol: [dropdown]
- tcpflags: tcpfin, tcpsyn, tcprst, tcpsh, tcpack, tcpurg
- exceptions: Non-IP, ARP, IP Opt, TTL
- aux: multicast:
- port/plugin selection: plugin only
- output ports: 2
- output plugins: 0
- qid: 300
- plugin tag(0-31 or *): 0
- priority(0-63): 50

The 'NPR.2:port2 Queues' dialog box is also open, showing the following configuration:

queue id	threshold(bytes)	quantum
default	32768	256
300	100000	1500

Port Rate(Mbps): 0.683

50 msec Delay (Default)

The screenshot shows the Remote Laboratory Interface (RLI) configuration for a network. On the left, a topology diagram shows two nodes, onl37 and onl32, connected. onl37 has IP Base Address: 192.168.2.32 and CP host: onl037.arl.wustl.edu. Below the topology is a diagram of the filter configuration: onl37 has ports 1, 2, 3, and 4. Port 1 is connected to a filter 'F' with a red 'D' and a green 'Q300' (Queue). Port 2 is connected to a filter 'F' with a red 'D'. Port 3 is connected to a filter 'F' with a red 'D'. Port 4 is connected to a filter 'F' with a red 'D'.

The terminal window shows the following output:

```

onlusr> ssh onl037
Last login: Wed Jan 28 10:22:24 2009 from onlusr.arl.wustl.edu
onl037> netstat -i
Kernel Interface table
Iface    MTU  Met  RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0     1500  0    4213   0     0     0     3552   0     0     0   0 BMRU
eth1     1500  0     0     0     0     0     3     0     0     0   0 BMRU
lo       16436 0    94     0     0     0     94    0     0     0   0 LRU
onl037> ping -c 5 n1p1
PING n1p1a (192.168.1.32) 56(84) bytes of data:
64 bytes from n1p1a (192.168.1.32): icmp_seq=1 ttl=62 time=257 ms
64 bytes from n1p1a (192.168.1.32): icmp_seq=2 ttl=62 time=50.1 ms
64 bytes from n1p1a (192.168.1.32): icmp_seq=3 ttl=62 time=50.1 ms
64 bytes from n1p1a (192.168.1.32): icmp_seq=4 ttl=62 time=50.1 ms
64 bytes from n1p1a (192.168.1.32): icmp_seq=5 ttl=62 time=50.1 ms
--- n1p1a ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 50.101/91.638/257.772/83.067 ms
onl037>
    
```

Send Control Message to Plugin

The screenshot displays a network router interface with several windows and a terminal. The 'Router NPR. 2 P...' window shows a menu with options like 'Send Command to Plugin' and 'Show Plugin Command Log'. The 'NPR. 2 Send Command CommandPlugin' dialog box is open, showing 'microengine: 0' and 'message: delay= 100'. A 'Plugin Command Log: delay microe...' window shows the output 'delay= 100 -- succeeded. Returned 100'. A terminal window shows a ping command being executed, with the output showing a delay of 100ms for each of the five packets. A yellow box on the right lists other messages: 'delay' (return delay in msec), 'counts' (return npkts, maxinq, ndrops), and 'reset' (reset global counters).

Router NPR. 2 P... Edit

- Send Command to Plugin
- Show Plugin Command Log
- Add Plugin Data to Graph
- Add Plugin Directory
- Delete Plugin
- Add Plugin

NPR. 2 Send Command CommandPlugin

microengine: 0
message: delay= 100

Enter Cancel

Plugin Command Log: delay microe...

delay= 100 -- succeeded. Returned 100

other messages:

- =delay return delay (msec)
- =counts return npkts, maxinq, ndrops
- reset reset global counters

```
onl
onl037> ping -c 5 nlp1
PING nlp1 (192.168.1.32) 56(84) bytes of data:
64 bytes from nlp1 (192.168.1.32): icmp_seq=1 ttl=62 time=100 ms
64 bytes from nlp1 (192.168.1.32): icmp_seq=2 ttl=62 time=100 ms
64 bytes from nlp1 (192.168.1.32): icmp_seq=3 ttl=62 time=100 ms
64 bytes from nlp1 (192.168.1.32): icmp_seq=4 ttl=62 time=100 ms
64 bytes from nlp1 (192.168.1.32): icmp_seq=5 ttl=62 time=100 ms

--- nlp1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3999ms
rtt: min/avg/max/mdev = 100.100/100.105/100.116/0.283 ms
onl037>
```