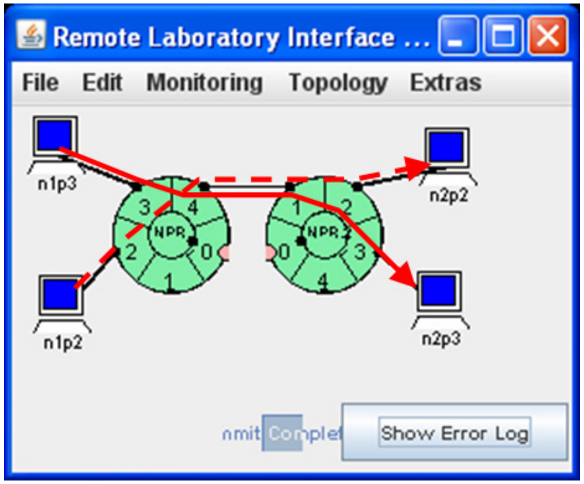
Open Network Laboratory



Raj Jain
Washington University in Saint Louis
Saint Louis, MO 63130
Jain@wustl.edu

Audio/Video recordings of this lecture are available on-line at:

http://www.cse.wustl.edu/~jain/cse473-11/

Overview

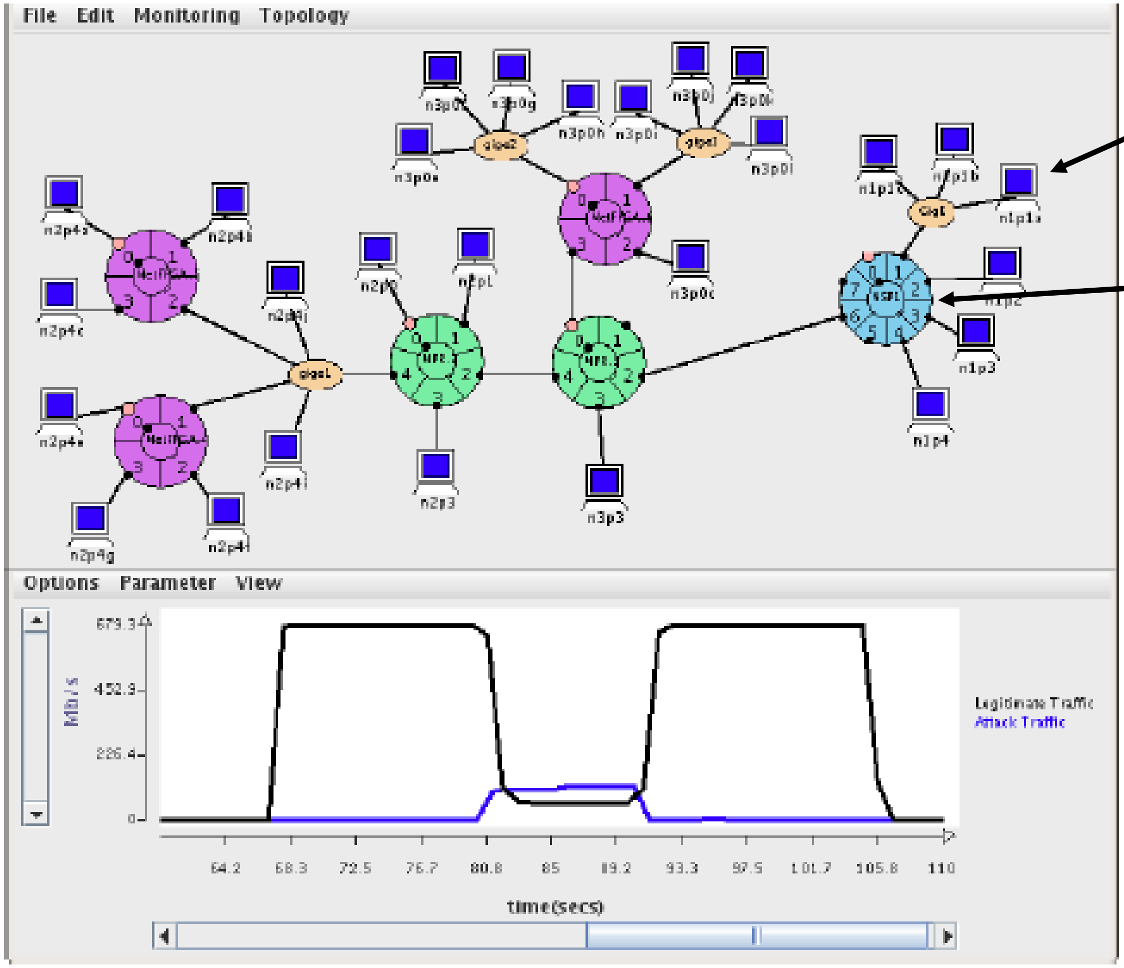
1. Open Network Laboratory (ONL)
2. Remote Laboratory Interface
3. Running ONL experiment
4. SSH Tunnel Configuration
5. Lab assignments

Note: These slides are based mostly on presentations available at ONL website.

Open Network Laboratory (ONL)

- Developed by Prof. Jon Turner and his team at WUSTL
- Allows students to set up networking configurations consisting of routers and hosts and experiment with them
- Allows real-time visualization of various queues and traffic flows
- Allows running programs on the hosts and filters on programmable routers
- Also useful for research on networking protocols and applications requiring multiple hosts
- All of the resources are available remotely for use by anyone. Any one can register and use.
- Ref: http://onl.wustl.edu

Sample Configuration

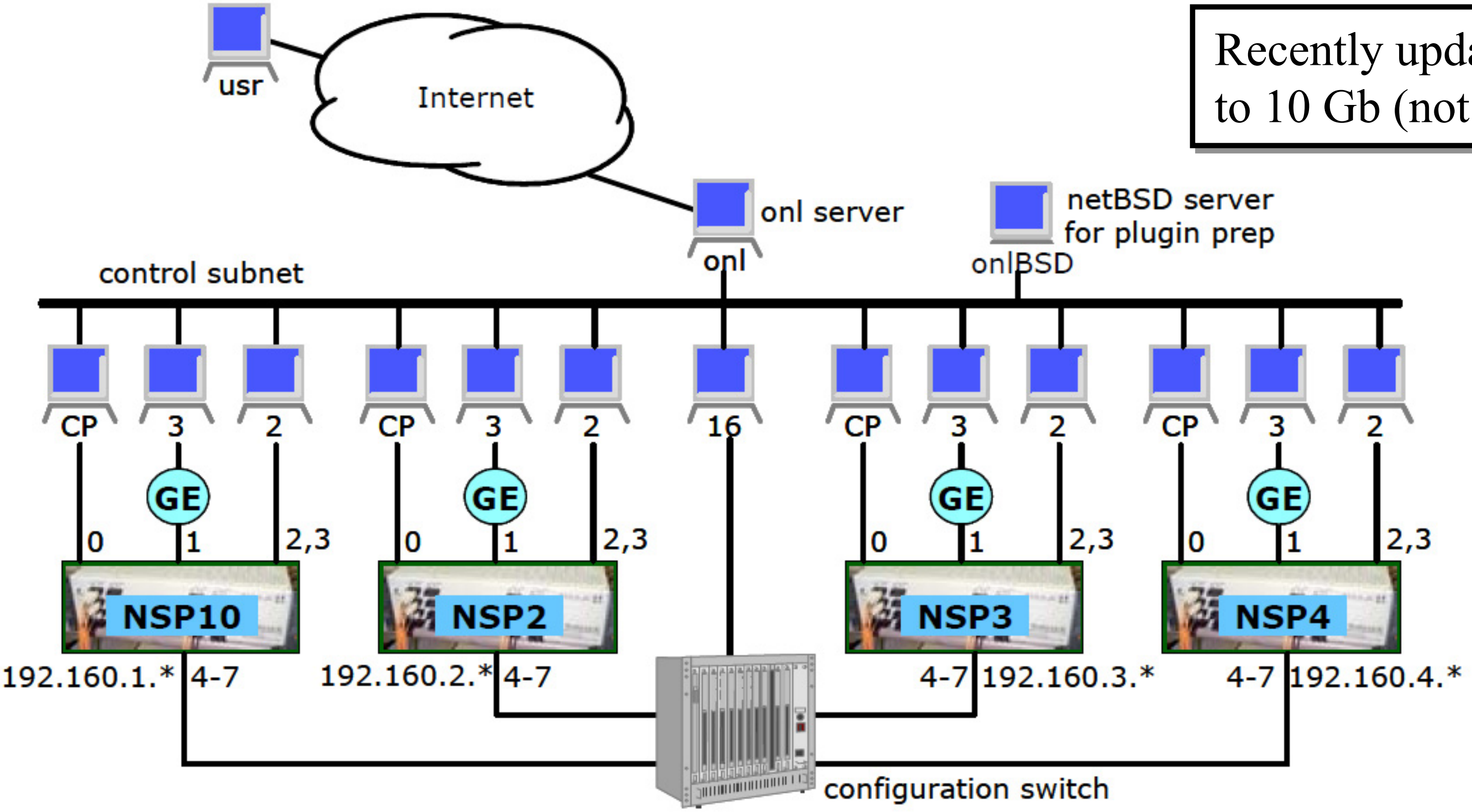


Hosts

Routers

Traffic monitoring

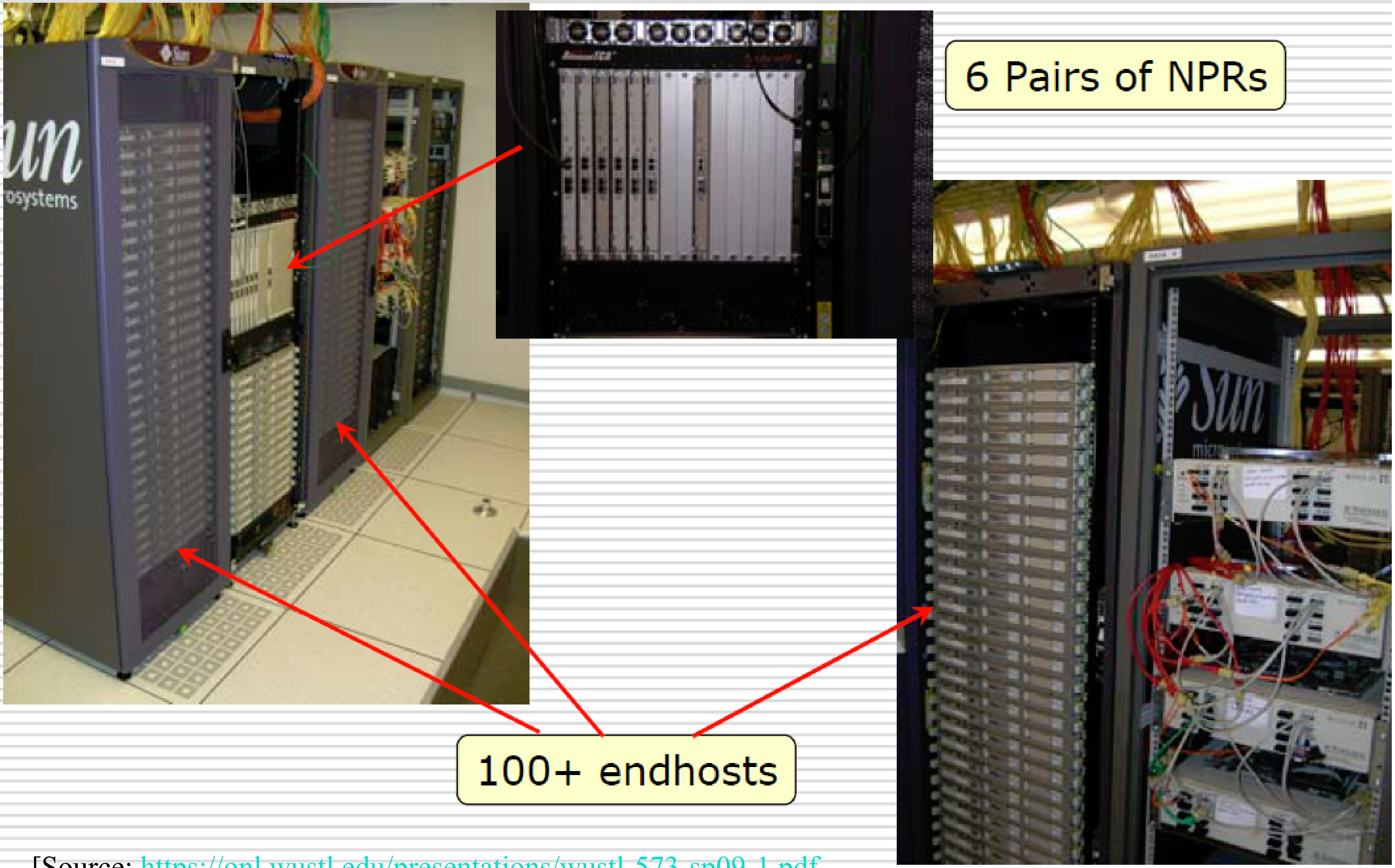
Physical ONL Hardware



Recently updated to 10 Gb (not shown)

- NSP = Network Service Platform (Router)
- CP = NSP Control Processor
- GE = Gigabit Ethernet Switch
- 100+ hosts
- Configuration Switch connects any router port to another router or host

Physical ONL Hardware (Cont)



6 Pairs of NPRs

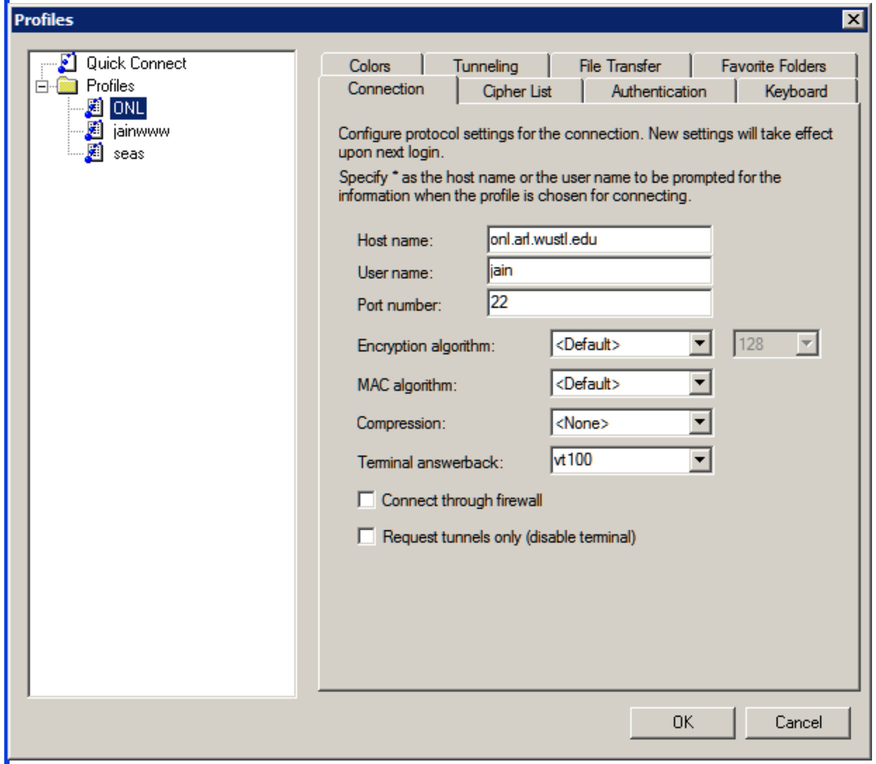
100+ endhosts

[Source: https://onl.wustl.edu/presentations/wustl-573-sp09-1.pdf

Using ONL – 7 Steps

1. Read ONL tutorial
    1. The ONL Tutorial, http://wiki.arl.wustl.edu/onl/index.php/The_ONL_Tutorial
    2. Remote Laboratory Interface, http://wiki.arl.wustl.edu/onl/index.php/Remote_Laboratory_Interface_%28RLI%29
    3. Getting started, https://onl.wustl.edu/restricted/getting-started.html
2. SSH (on MACs and Linux) or Windows Putty, http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
3. Install Java Runtime Environment (JRE) V1.6 or higher, Check "java –version", if necessary download from http://java.com/en/download/manual.jsp
4. Download RLI.Jar, https://onl.wustl.edu/restricted/export/RLI.jar
5. SSH to onl.wustl.edu
6. Run RLI, Prepare your configuration
7. Reserve time, Commit and Run.
   Commit again after any topology modification.

SSH



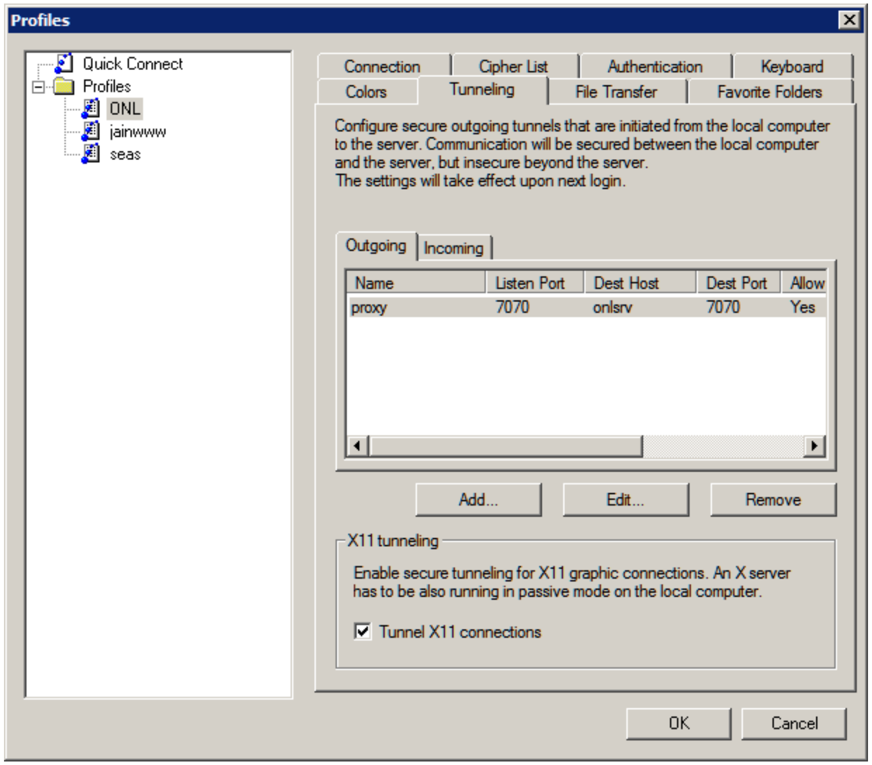
Configure protocol settings for the connection. New settings will take effect upon next login.

Specify * as the host name or the user name to be prompted for the information when the profile is chosen for connecting.

Host name:	onl.arl.wustl.edu
User name:	jain
Port number:	22
Encryption algorithm:	<Default> 128
MAC algorithm:	<Default>
Compression:	<None>
Terminal answerback:	vt100

□ Connect through firewall

□ Request tunnels only (disable terminal)



Configure secure outgoing tunnels that are initiated from the local computer to the server. Communication will be secured between the local computer and the server, but insecure beyond the server.
The settings will take effect upon next login.

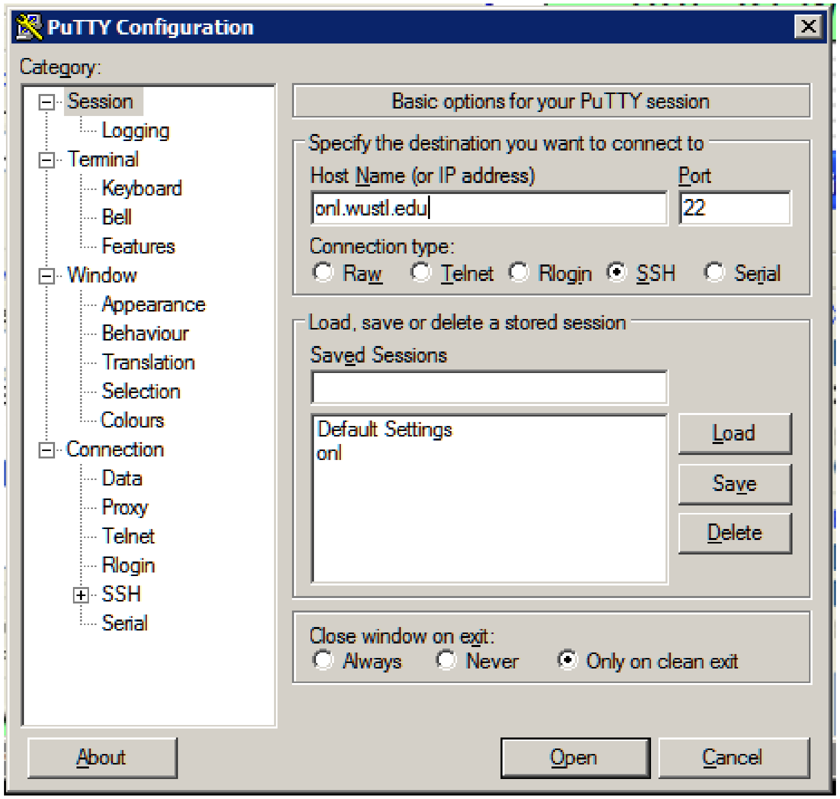
Name	Listen Port	Dest Host	Dest Port	Allow
proxy	7070	onlsrv	7070	Yes

X11 tunneling

Enable secure tunneling for X11 graphic connections. An X server has to be also running in passive mode on the local computer.

☑ Tunnel X11 connections

Putty



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): onl.wustl.edu

Port: 22

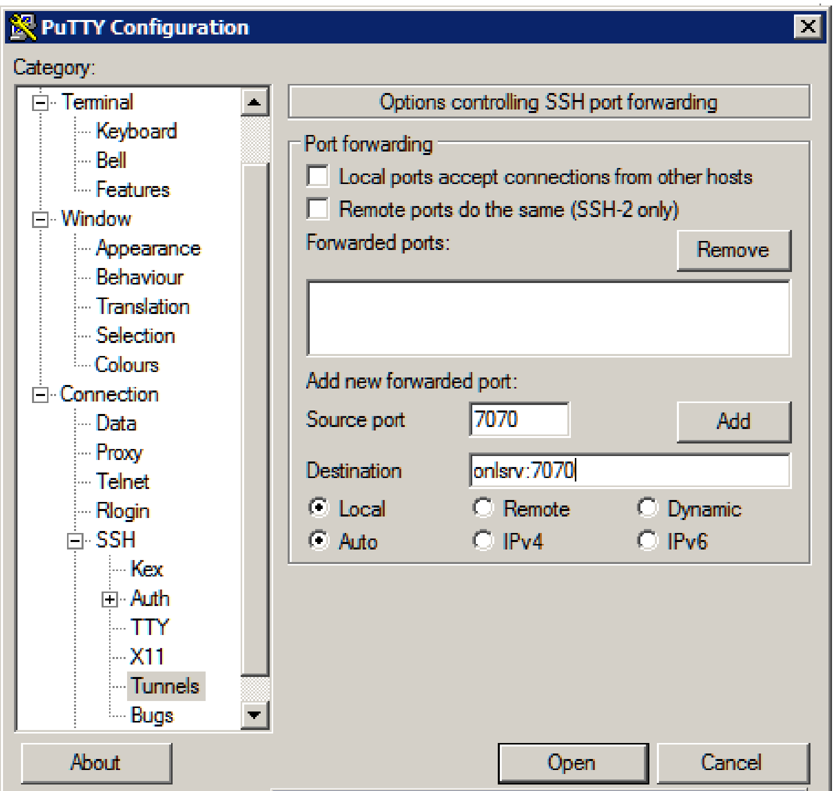
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions:

- Default Settings
- onl

Close window on exit: ○ Always ○ Never ● Only on clean exit



PuTTY Configuration

Options controlling SSH port forwarding

Port forwarding

☐ Local ports accept connections from other hosts

☐ Remote ports do the same (SSH-2 only)

Forwarded ports:

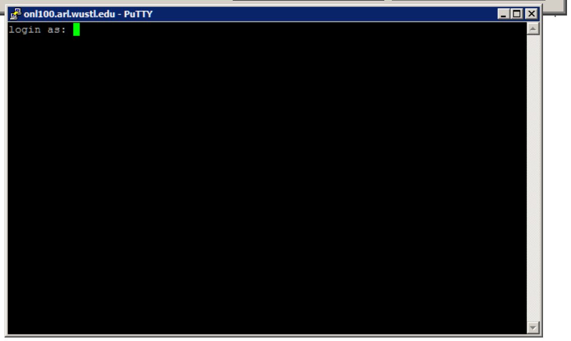
Add new forwarded port:

Source port: 7070

Destination: onlsrv:7070

● Local ○ Remote ○ Dynamic

● Auto ○ IPv4 ○ IPv6



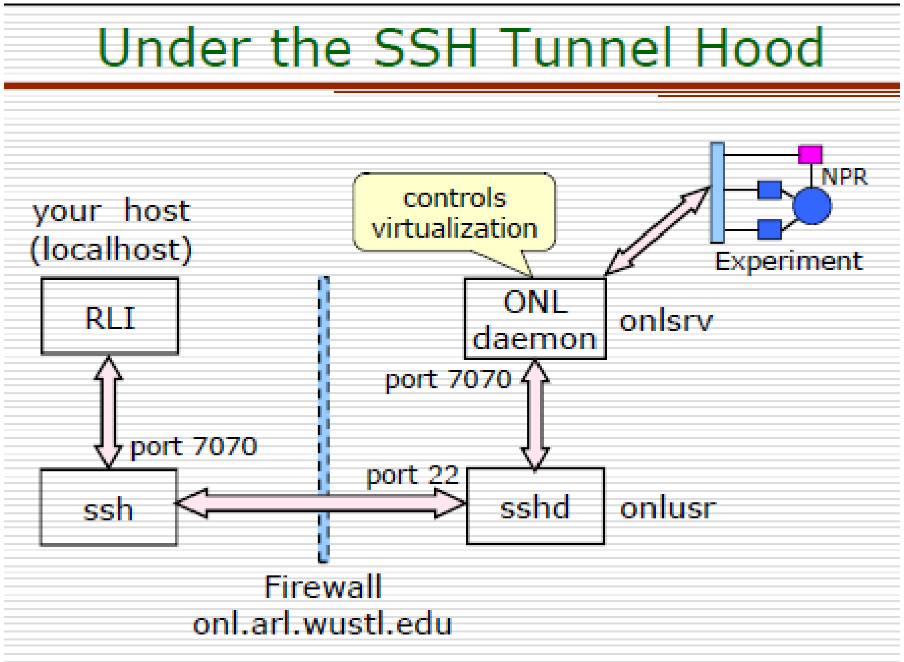
onl100.arl.wustl.edu - PuTTY

```
login as:
```

- SSH –
- This will open a SSH window. Login.

[Source: https://onl.wustl.edu/presentations/wustl-573-sp09-1.pdf]

SSH Tunnel Configuration

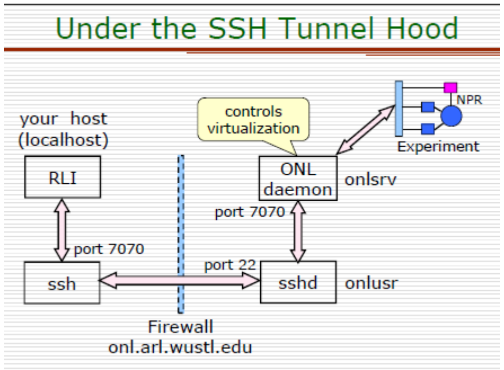


- Build before each experimental session
- Allows your RLI to communicate with ONL daemon
- Needed to make reservation and commit
- SSH tunneling

  Unix command line

  - ssh -L 7070:onlsrv:7070 username@onl.arl.wustl.edu
  - Windows PuTTy
  - Windows SSH client

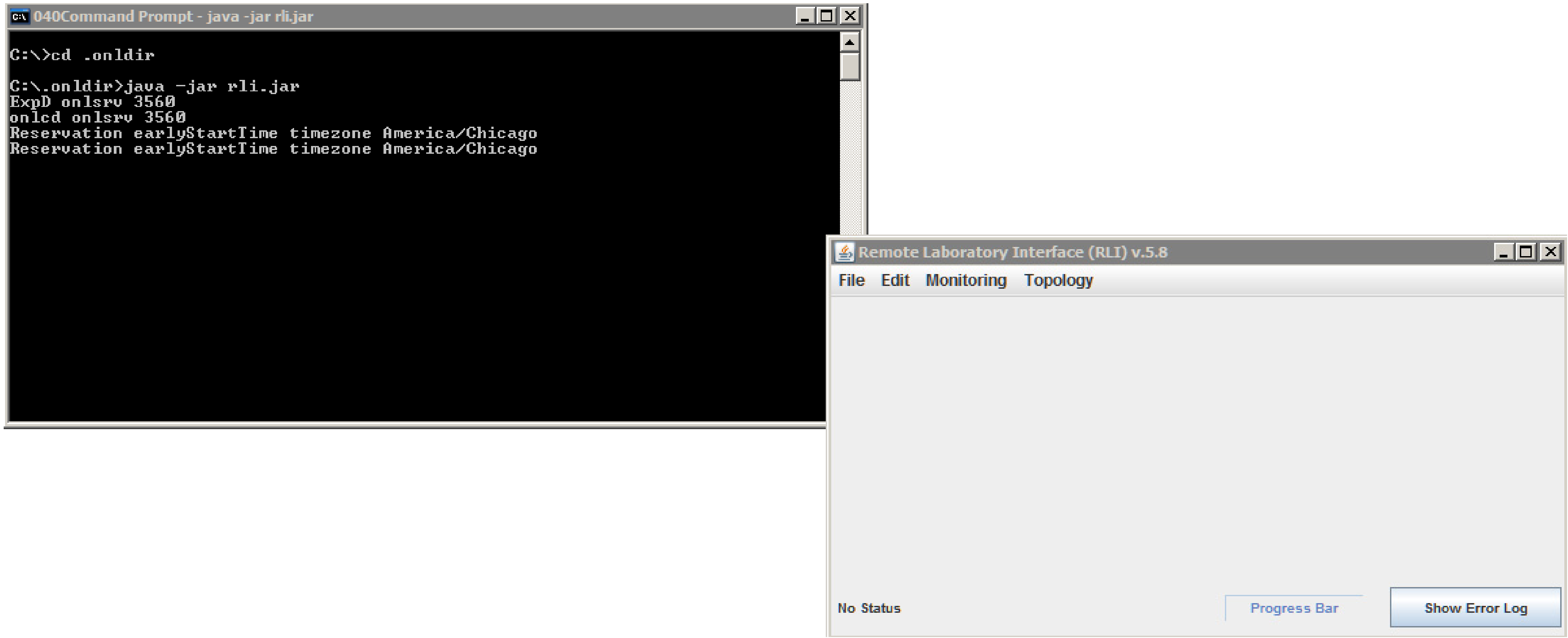
ONL SSH Restrictions


Under the SSH Tunnel Hood

- You can login into ONL hosts in your configuration
- Same password for ONL host login and Web login
- You can SSH from one ONL host to another
  Password-free SSH between ONL hosts
- You can only access ONL hosts assigned to your experiment
- From your PC, you can only SSH to onl.wustl.edu
  Gets connected to onlusr – the ONL user host
- Firewall blocks all connections from within ONL to outside
- You can pull (save) from ONL host to your PC
  You can not save in ONL host from your PC
- You can push (open) to ONL host from your PC
  You can not open a ONL host file from your PC

Remote Laboratory Interface

- Using a Java **RLI** you can configure and run experiments from your computer using a **SSH tunnel**
- On your computer (not SSH window)
  - cd c:\.onldir
  - java –jar rli.jar

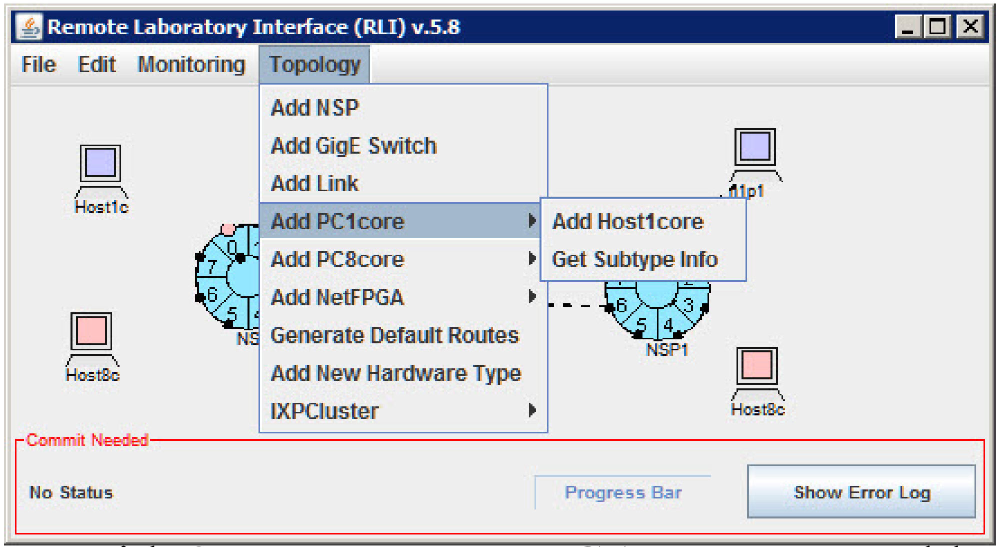


```
C:\>cd .onldir

C:\.onldir>java -jar rli.jar
ExpD onlsrv 3560
onlcd onlsrv 3560
Reservation earlyStartTime timezone America/Chicago
Reservation earlyStartTime timezone America/Chicago
```

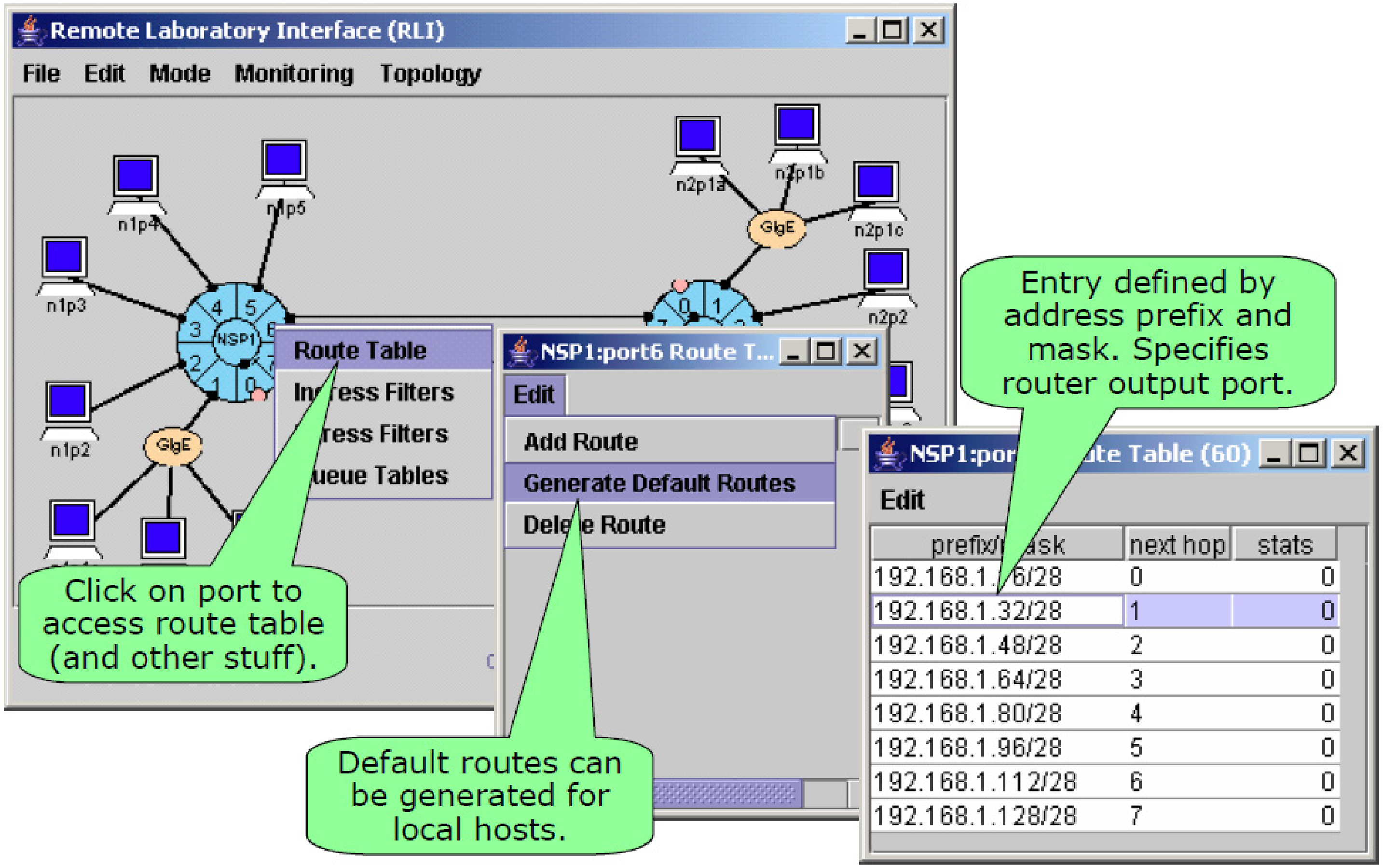
040Command Prompt - java -jar rli.jar
Remote Laboratory Interface (RLI) v.5.8
File
Edit
Monitoring
Topology
No Status
Progress Bar
Show Error Log

RLI (Cont)



- NSP: IPv4 Router with 8 ×1GbE
- GigE Switch: 1GbE and 10GbE
- Link
- PC1Core: Single core PC with 1 GbE
- PC8Core: 8-core PC with 10GbE
- NetFPGA: Programmable FPGA
- Generate Default Routes
- New Hardware type
- IXPCluster: Intel Network Processor (IXP 2800) based programmable routers

RLI (Cont)



prefix/mask	next hop	stats
192.168.1.16/28	0	0
192.168.1.32/28	1	0
192.168.1.48/28	2	0
192.168.1.64/28	3	0
192.168.1.80/28	4	0
192.168.1.96/28	5	0
192.168.1.112/28	6	0
192.168.1.128/28	7	0

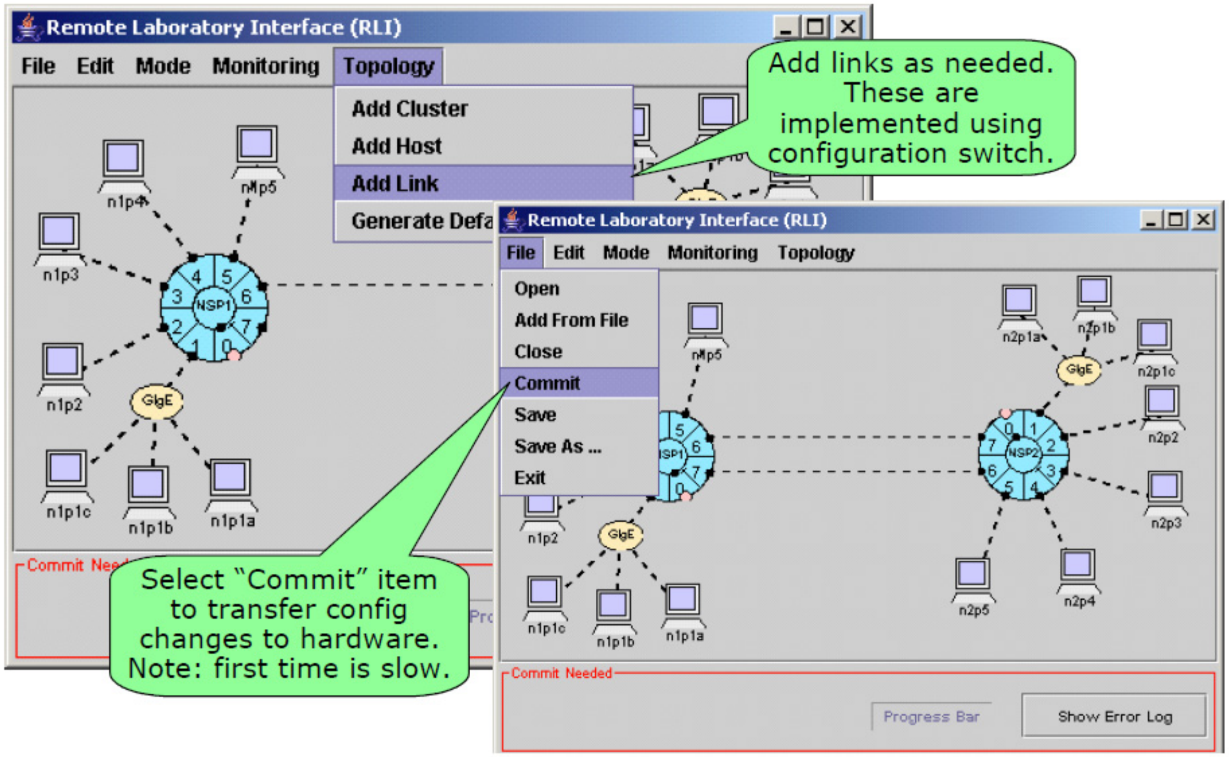
Entry defined by address prefix and mask. Specifies router output port.

Click on port to access route table (and other stuff).

Default routes can be generated for local hosts.

RLI (Cont)

- To run the experiment, you need to **commit** your configuration to physical hardware
  Only one experiment at a time ⇒Need to **reserve**
- No reservation required for designing the experiment (setting the configuration etc)

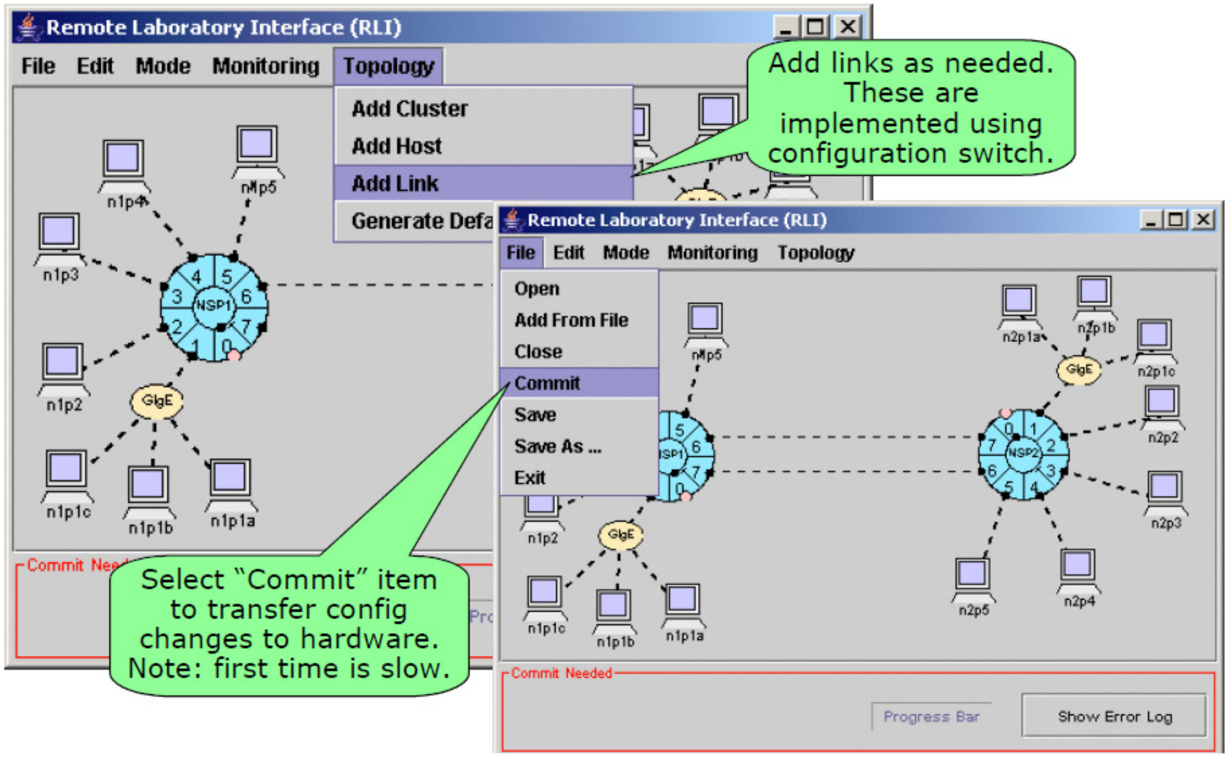


Add links as needed. These are implemented using configuration switch.

Select "Commit" item to transfer config changes to hardware. Note: first time is slow.

RLI (Cont)

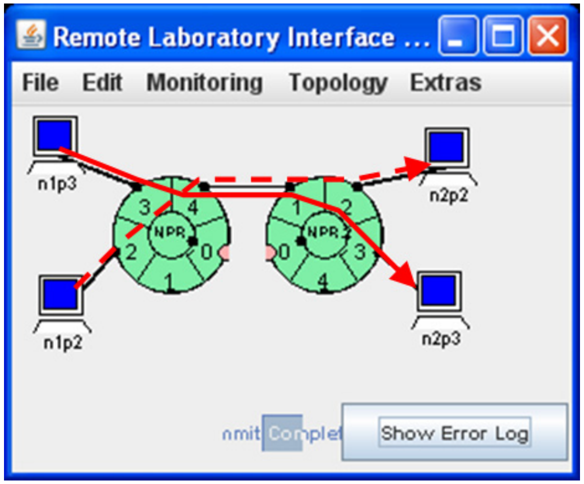
- To run the experiment, you need to **commit** your configuration to physical hardware
  Only one experiment at a time ⇒Need to **reserve**
- No reservation required for designing the experiment (setting the configuration etc)



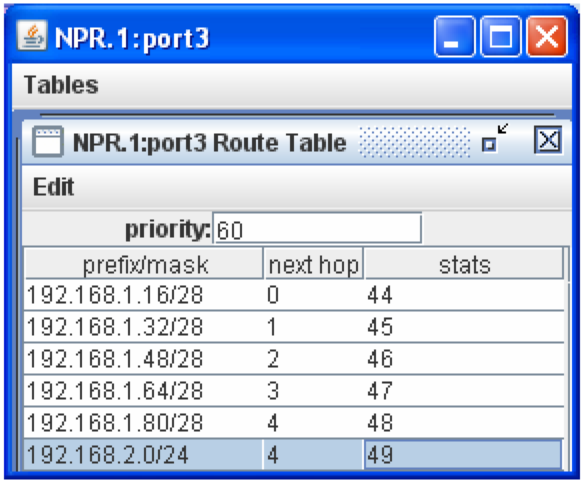
Add links as needed. These are implemented using configuration switch.

Select "Commit" item to transfer config changes to hardware. Note: first time is slow.

RLI (Cont)



Configure network topology



NPR.1:port3

Tables

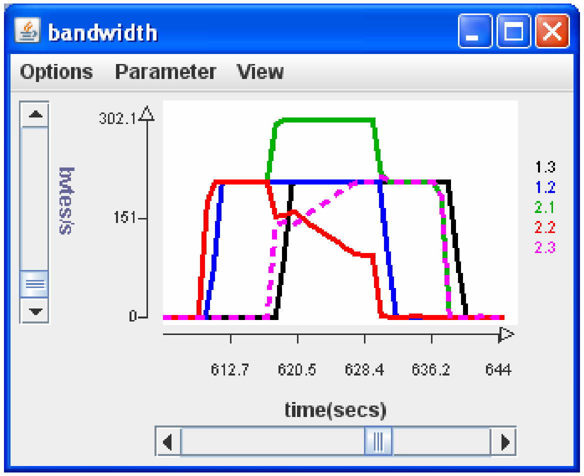
NPR.1:port3 Route Table

Edit

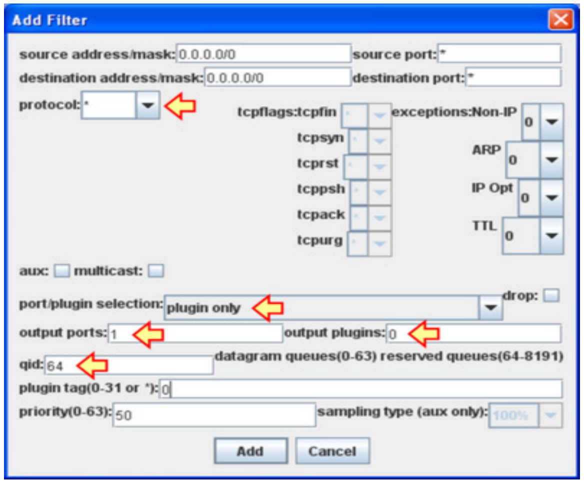
priority: 60

prefix/mask	next hop	stats
192.168.1.16/28	0	44
192.168.1.32/28	1	45
192.168.1.48/28	2	46
192.168.1.64/28	3	47
192.168.1.80/28	4	48
192.168.2.0/24	4	49

Routing and forwarding



Real-time charts



Add Filter

source address/mask: 0.0.0.0/0 source port: *

destination address/mask: 0.0.0.0/0 destination port: *

protocol: *

tcpflags:tcpfin ·

tcpsyn ·

tcprst ·

tcppsh ·

tcpack ·

tcpurg ·

exceptions:Non-IP 0

ARP 0

IP Opt 0

TTL 0

aux: □ multicast: □

port/plugin selection: plugin only

drop: □

output ports: 1 output plugins: 0

qid: 64 datagram queues(0-63) reserved queues(64-8191)

plugin tag(0-31 or *): 0

priority(0-63): 50 sampling type (aux only): 100%

Add Cancel

Adding filters

iPerf

- A tool to send UDP or TCP traffic between two nodes
- Versions:
  - iPerf v3: http://code.google.com/p/iperf/downloads/list
  - iPerf v2: http://sourceforge.net/projects/iperf/?abmode=1
  - iPerf v1: https://publishing.ucf.edu/sites/itr/cst/Pages/IPerf.aspx (windows binary executable)

Ref: http://en.wikipedia.org/wiki/Iperf, http://linhost.info/2010/02/iperf-on-windows/

iPerf (cont)

```
E:\m>iperf -h
Usage: iperf [-s|-c host] [options]
       iperf [-h|--help] [-v|--version]

Client/Server:
  -f, --format    [kmKM]   format to report: Kbits, Mbits, KBytes, MBytes
  -i, --interval  #        seconds between periodic bandwidth reports
  -l, --len       #[KM]    length of buffer to read or write (default 8 KB)
  -m, --print_mss          print TCP maximum segment size (MTU - TCP/IP header)
  -o, --output    <filename> output the report or error message to this specifie
d file
  -p, --port      #        server port to listen on/connect to
  -u, --udp                use UDP rather than TCP
  -w, --window    #[KM]    TCP window size (socket buffer size)
  -B, --bind      <host>   bind to <host>, an interface or multicast address
  -C, --compatibility      for use with older versions does not sent extra msgs
  -M, --mss       #        set TCP maximum segment size (MTU - 40 bytes)
  -N, --nodelay            set TCP no delay, disabling Nagle's Algorithm
  -V, --IPv6Version        Set the domain to IPv6
```

iPerf (Cont)

Server specific:

-s, --server run in server mode
-D, --daemon run the server as a daemon
-R, --remove remove service in win32

Client specific:

-b, --bandwidth #[KM] for UDP, bandwidth to send at in bits/sec
(default 1 Mbit/sec, implies -u)
-c, --client <host> run in client mode, connecting to <host>
-d, --dualtest Do a bidirectional test simultaneously
-n, --num #[KM] number of bytes to transmit (instead of -t)
-r, --tradeoff Do a bidirectional test individually
-t, --time # time in seconds to transmit for (default 10 secs)
-F, --fileinput <name> input the data to be transmitted from a file
-I, --stdin input the data to be transmitted from stdin
-L, --listenport # port to recieve bidirectional tests back on
-P, --parallel # number of parallel client threads to run
-T, --ttl # time-to-live, for multicast (default 1)

iPerf (Cont)

```
Miscellaneous:
  -h, --help              print this message and quit
  -v, --version             print version information and quit

[KM] Indicates options that support a K or M suffix for kilo- or mega-

The TCP window size option can be set by the environment variable
TCP_WINDOW_SIZE. Most other options can be set by an environment variable
IPERF_<long option name>, such as IPERF_BANDWIDTH.

Report bugs to <dast@nlanr.net>
```

iPerf Command Examples

- **iperf -s –u**
  Get ready to receive (server) UPD traffic
- **iperf -c n2p2 –u -b 10m -t 20**
  Send (client) to n2p2 UDP traffic at 10 Mbps for 20s
- **iperf -s -w 4m**
  Get ready to receive TCP traffic with a socket buffer of 4 MB
  Window = 2 × Socket Buffer
- **iperf -c n2p2 -w 3m -t 20**
  Send to n2p2 TCP traffic with a socket buffer of 3 MB for 20s
- Note:
  - Storage: 1MB = 1024 KB = 2^{20} B
    Big K = 1024, Big B=Bytes
  - Networking: 1Mb= 1000 kb = 10^6 b (not 2^{20} b)
    Little k = 1000, Little b = bits

Selective Acknowledgement

- Destinations can indicate exactly which packets are missing
- Useful if long-delay or high-speed (large Window)
- TCP Segment Format:

Source Port	Dest Port	Seq No	Ack No	Data Offset	Resvd	U	A	P	R	S	F
16	16	32	32	4	6	1	1	1	1	1	1

Window	Check-sum	Urgent	Options	Pad	Data
16	16	16	x	y	← Size in bits

- TCP Connection Setup: 2-byte SACK permitted option

Kind=4	Length=2
8b	8b

- Sack option in Data Packets:

Kind=5	Length =8*n*+2	Left Edge of 1st Block	Right Edge of 1st Block	LE of 2nd Block	RE of 2nd Block	…	LE of *n*th Block	RE of *n*th Block
8b	8b	32b	32b	32b	32b		32b	32b

Ref: http://tools.ietf.org/html/rfc2018

Lab Assignments

Objective: Hands-on experience & apply concepts

Lab assignment 1:

- Familiarize with ONL through ONL tutorial
- Network topology, packet path (forwarding), link capacity

Lab assignment 2:

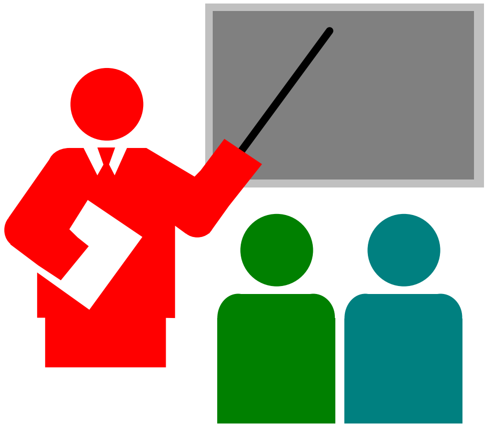
- Transmit and monitor packet traffic
- Routing (edit routing table)
- Analyze behavior of a queue

Lab assignment 3:

- Congestion Control

Note: It is important for each student to do the labs individually. ONL keeps track of who did what.

Summary



1. Open Network Laboratory (ONL) allows remote users to setup a network configuration and experiment
2. Remote Laboratory Interface (RLI) is a java frontend for designing and running experiments
3. Need to setup SSH tunnel to ONL server
4. Need to reserve the physical equipment before committing your experiment to hardware
5. Recommend using during working hours to avoid crashed systems and other problems