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

Network Configuration

Routing Tables

Bandwidth Usage

Queue Length

Queue Parameters
People Who Make it Happen

Jon Turner
Principal 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

Gigabit Router

Smart Port Card

Field Prog. Port Extender
Basic NSP Architecture

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

control network

experiment network

configuration switch

YOU

Internet

onl logins

onl server

onlusr

onlsrv

CP

GE

NSP1

NSP2

NSP3

NSP4

0

1

2,3

0

1

2,3

0

1

2,3

0

1

2,3

4-7

4-7

4-7

4-7

C树

配置开关
Major Software Components

- Remote Lab Interface (RLI)
- SSH tunnel
- SSH logins
- ONL daemon and SSH proxy
- Host daemon
- Configuration switch

You

Internet

ONL server

ONL logins

ONL daemon

and SSH proxy

YOU

switch controller

and SPC control

message handler

configuration switch
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

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

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

Add hosts and links as needed.

Drag icons to improve visual layout.

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

For all ports

**Remote Laboratory Interface (RLI)**
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

chart name

poll every 0.3 sec
Ping Chart

- Select max y-value = 0.004
- Window width halved
Adding A Link

drag from port 7 to port 6
Delete Route Group

- Select the routes to delete.
- Press shift+select to delete the specified routes.
- The route deleted message appears.

Example routes for deletion:
- 192.168.1.16/28
- 192.168.1.128/28
Add Route To Loopback

match n1p2 address

route added
Demo 3

- Traffic monitoring
- iperf traffic generator
- Link capacity
Bandwidth Monitoring

![Bandwidth Monitoring Interface](image)

- **sndr**
- **rcvr**
Generating Traffic with Iperf

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

Available at [http://dast.nlanr.net/projects/iperf/](http://dast.nlanr.net/projects/iperf/), installed on all onl hosts
iperf Traffic Generator

Start UDP server (receiver)

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

- 150 Mbps each flow
- Q300 and Q301 filled
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