PERT OPERATIONS

Training Workbook

PERT OPERATIONS Training Workbook

Issue 1.0

Table of Contents

Case Study: Finding out how to Tune Your Laptop's TCP Stack

Firstly, check your computer's current configuration.

Examine the information in the PERT Knowledge Base (PERT KB) about your operating system. (see **http://kb.pert.geant2.net/PERTKB/EndSystemTuning**).

Using this information, find out how much throughput you can expect on a path with 100ms RTT in the best case (transmit and receive).

How would you tune your laptop to sustain 100Mbps at 120ms RTT (send and receive)?

Case Study: Running a UDP iperf Test

Running a UDP iperf test with 1 Mbit/s for 15 seconds from your workstation to udp-iperf.switch.ch.

You can choose from the following iperf options:

- -s run in server mode
- -c run in client mode
- -u switch to UDP mode (default is TCP)
- -b sending rate for UDP mode in bits/sec
- -i reporting interval in seconds
- -w set TCP window size
- -t set test duration in seconds
- -P set number of parallel transfers (TCP only)

What useful information can you see from the test output?

Please note, that the only useful information that you get from iperf is the information from the Server Report (Server side is the receiver). Only the receiver can report about packet losses.

If you want to see more details about the distribution of errors or jitter, you have to run the server side with the option –i 1, which will report statistics every 1 second. Unfortunately you will only see the summary on the client side, so you will need access to the server side to see this enhanced statistics.

Case Study: Running a TCP iperf Test

Run a TCP iperf test with a single stream with a buffer size of 8K for 30 seconds with a reporting interval of 1 second from your workstation to tcp-iperf.switch.ch.

Try to vary the buffer sizes from 8K, 16K, 32K and 64K and see what happens to the throughput rates.

Try the following commands:

- iperf –c tcp-iperf.switch.ch –t 30 –i 1 –w 8K
- iperf –c tcp-iperf.switch.ch –t 30 –i 1 –w 16K
- iperf –c tcp-iperf.switch.ch –t 30 –i 1 –w 32K
- iperf –c tcp-iperf.switch.ch –t 30 –i 1 –w 64K

Now run a TCP iperf test with the standard parameters and more than 1 simultaneous stream and note what happens to the throughput rates.

Try the following commands:

- iperf –c tcp-iperf.switch.ch –P 2
- iperf –c tcp-iperf.switch.ch –P 3
- iperf –c tcp-iperf.switch.ch –P 4

Note that whenever possible, you should look at the test results from the server side, because the sender usually does not provide reliable information !For example, the duration of the test can be longer on the receiver side, because the sender side closes it's end as soon as it has given the last bytes to the TCP stack. Due to buffering it can take a while until all the bytes are really transferred to the receiver.

Exercise: Using an NDT Server

Using NDT to Help you Tune Performance

1. Ensure that your web browser supports Java 1.4.
2. Connect to **http://ndt.switch.ch**.
3. Click **<START>** to begin a test.

If someone else is already running a test, you will be put in a queue and your test will start automatically as soon as the other test has ended.

4. An applet will run a TCP transfer from your workstation to the SWITCH server and vice versa. The results should appear in about 30 seconds.

What are the speeds you can achieve in the inbound and outbound directions?

Inbound speed:

Outbound speed:

5. Click **<Statistics>**.

What tips can you find to increase your buffers?

Recommended receive buffers:

6. Click **<More Details…>** to open the Web100 Variables Java Applet Window.

What is the TCP buffer size of the NDT server?

7. Try to tune your TCP buffers to increase performance.

*You can find hints on how to do this at **http://kb.pert.geant2.net/PERTKB/OperatingSystemSpecific**.*

8. Run a further NDT test to determine whether your performance has improved.

What are the speeds you can now achieve in the inbound and outbound directions?

Inbound speed:

Outbound speed:

Key Information: TCPDUMP Command Line and Filtering Options

Useful tcpdump command line options:

- -i on which interface traffic should be captured
- -s number of bytes that should be captures (default is 96bytes, 0 means full packet)
- -w write to file
- -r read from file
- -c number of packets to capture

Filtering options:

• host	hostname	limits packets to and from this host
• udp		only udp packets
• tcp		only tcp packets
• port	portnumber	only in combination with udp or tcp
• icmp		only ICMP packets

Case Study: Using TCPDUMP to Capture ICMP Packets

Run tcpdump in one terminal and restrict it to only capture ICMP packets. Open another terminal and ping **www.switch.ch**.

You should see the ICMP echo requests and replies. When finished, end the tcpdump with Control-C.

Case Study: Using TCPDUMP and Wireshark Together

Run tcpdump in one terminal and restrict it to destination **www.switch.ch**. Activate capturing to a file called **traceroute.pcap**.

Open another terminal and traceroute to **www.switch.ch**.

When finished, end the tcpdump with **Control-C**.

Start **wireshark** and open the capture file that you just created.

Case Study: Identifying the Effects of a Middlebox

Open the prepared capture files **middlebox-a.pcap** and **middlebox-b.pcap**. These files contain a simple http request to a webserver behind a Firewall which is inspecting and modifying packets.

The **middlebox-a.pcap** file illustrates the way the client sees the transfer, whilst **middlebox-b.pcap** is the tcpdump of the server side of the connection. The two files show the same transaction viewed from both ends of a connection. Normally both captures should be identical. However the middlebox can introduce changes to the TCP stream.

Identify the changes that occur due to the middlebox and note them below.