Exercises: Basics of Transport Layer

Experiential Learning Workshop

1 General Guidelines

1. Make a team of four unless stated otherwise.
2. For each exercise, use `tcpdump` capture to verify contents
3. Ensure to use proper capture filter and don't capture irrelevant traffic
4. Where appropriate or applicable, use `wget` or `nc` to access the web server.
5. To kill any program in the Linux terminal, please press **`Ctrl-C`** and not **`Ctrl-z`**. The latter will suspend the program and not stop it.

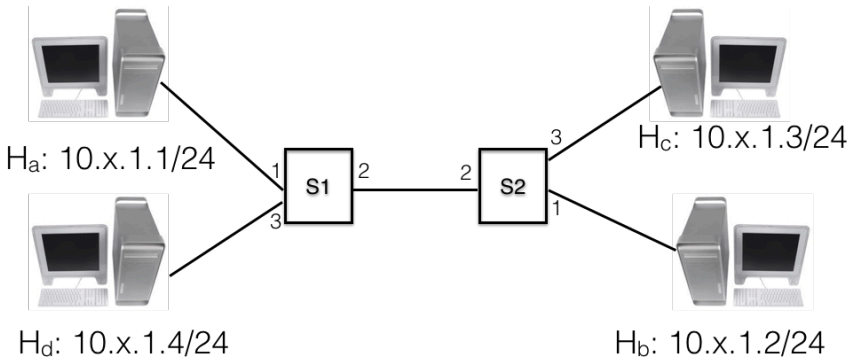
Note: Appendix provides instructions on installing any package if not already installed.

2 Configuring IP Addresses of machine

1. Use the network setting to set the manual network address i.e. replace DHCP configuration to manual configuration.
2. Use the IP addresses, subnet mask as shown in the diagram for the experiment. The default subnet mask is /24 or 255.255.255.0. Use the team number as value of x in 10.x.1.1, for first team, use 10.1.1.1, for 2nd team, use 10.2.1.1 etc.
3. Ensure you are not connected to college network. This is to ensure that there is no looping in the network otherwise, network can go down.

3 TCP Streaming, Reliability and UDP message boundary

The setup for this exercise requires two switches to be used as follows:



Note:

i. Configure the IP addresses as shown for four machines. Ensure IP addresses are configured manually and not via DHCP.

ii. We will use following simple python programs. You don't really need to know the python language syntax (reading of the program itself should explain what is being done).

```
a. udp_client.py
b. udp_server.py
c. tcp_client.py
d. tcp_server.py
```

i. Use the option -h to understand the usage syntax. By default, server listens on all the IP address and port number 9999 and receive in the buffer size of 10 bytes and reads data from socket every 1 second and displays the same. The client program by default connects to server on port 9999, but requires server IP to be specified, the client uses a default buffer size of 100 bytes and sends 10 messages at interval of 5 seconds each. The options for these programs are
   a. –s <server IP address>
   b. –p <server port number>
   c. –b <buffer size>
   d. –c <count of packets/messages to be sent>. The first message contains sequence of A… characters, 2nd message contains sequence of B… characters, and so on.
   e. –d delay (in sending by client or receiving by server)

3.1 UDP message Orientation

1. On H2, run the server `./udp_server.py` with buffer size of 20.
2. On H1, run the client `./udp_client.py` sending the message to server with buffer size of 50 bytes.
3. Note down what is displayed by the server. Explain why the server does not receive full packet. Change the buffer size of server to study further.
4. Analyze wireshark capture on what is being transmitted by client.

3.2 UDP Packet Loss

1. Repeat the above exercise with client sending 10 packets at interval of 5 seconds.
2. Break the link between switch S1 and S2 at 12th seconds and restore this link at 26th second. To break the link, just remove the wire between 2 switches or between switch and the wall jack.
3. Using wireshark, analyze the packet sent by client as well as those received by server. Explain which packets are received and which are lost.

3.3 TCP Stream based transmission

1. Repeat the above exercise but with TCP client and servers.
2. Analyze if the server receives full data.
3. Explain and understand the difference between TCP and UDP behavior.

3.4 TCP Packet Loss and Recovery

1. Repeat the above exercise.
2. Analyze if the server receives full data i.e. data transmitted during the time when link between S1 and S2 is broken/disconnected.
3. Analyze how many TCP segments were transmitted vs how many received.
4. Analyze if retransmitted segment contained any new data?
5. Analyze the time difference between retransmitted segments.
6. Explain and understand the difference between TCP and UDP behavior.

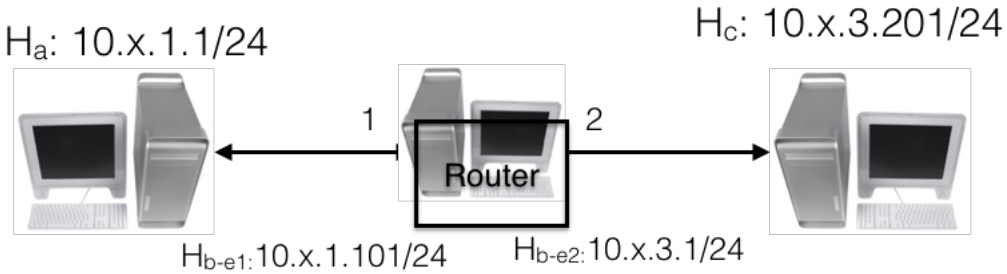
3.5 Exercise Expectations

1. Ability to understand UDP message delivery being boundary oriented.
2. Ability to understand TCP based message streaming i.e. no message boundaries.

3. Ability to under packet loss recovery in TCP whereas there is no recovery in UDP packet loss.

4 IP Routing

4.1 Basic Routing



1. Connect 3 m/cs in a network as shown in diagram. Assign the address manually to these systems Use the appropriate mask. Use value of x as per your team number e.g. 1 for team 1, 2 for team 2 and so on. To get a second interface on B, use the USB to ethernet adaptor on B.

2. Define IP addresses for B. Assign the IP addresses via the network Setting UI or via the following command.
   a. `sudo ip addr add 10.x.1.101/24 dev eno1`
   b. `sudo ip addr add 10.x.3.1/24 dev enx...` (USB device)

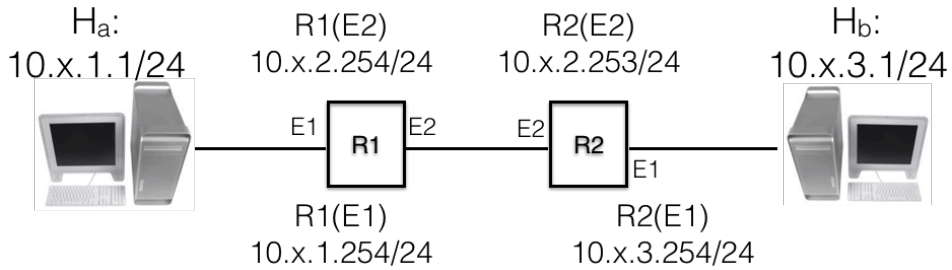
3. Convert machine B into a router
   a. `sudo sysctl -w net.ipv4.ip_forward=1`

4. Define routing for network 10.1.3.0/24 on **Ha**.
   a. `sudo ip route add 10.x.3.0/24 via 10.x.1.101`

5. Define routing for network 10.1.1.0./24 on **Hc**
   a. `sudo ip route add 10.x.1.0/24 via 10.x.3.1`

6. Check reachability between **Ha** and **Hc** (use ping to check)

4.2 Multi Network Routing



1. Connect 4 m/cs in a network as shown in diagram. Assign the address manually to these systems Use the appropriate mask. Use value of x as per your team number e.g. 1 for team 1, 2 for team 2 and so on. To get a second interface on B, use the USB to ethernet adaptor on B.

2. Define IP addresses for R1. Assign the IP addresses via the network Set it via UI or via the following command, and convert it into a router, and define routing table entry for 10.x.3.0/24

```
a. sudo ip addr add 10.x.1.254/24 dev eno1
b. sudo ip addr add 10.x.2.254/24 dev enx… (USB device)
c. sudo sysctl -w net.ipv4.ip_forward=1
d. sudo ip route add 10.x.3.0/24 via 10.x.2.253
```

Define IP addresses for R2. Assign the IP addresses via the network Set it via UI or via the following command, and convert it into a router, and define routing table entry for 10.x.3.0/24

```
a. sudo ip addr add 10.x.2.253/24 dev eno1
b. sudo ip addr add 10.x.3.254/24 dev enx… (USB device)
c. sudo sysctl -w net.ipv4.ip_forward=1
d. sudo ip route add 10.x.1.0/24 via 10.x.2.254
```

3. Define routing for 2nd and 3rd network on Ha,

```
a. sudo ip route add 10.x.2.0/24 via 10.x.1.254
b. sudo ip route add 10.x.3.0/24 via 10.x.1.254
```

4. Define routing for 1st and 2nd network on Hb,

```
a. sudo ip route add 10.x.1.0/24 via 10.x.3.254
b. sudo ip route add 10.x.2.0/24 via 10.x.3.254
```

5. Check reachability between **Ha** and **Hb** (use ping to check)

4.3 TTL Expiry

1. Connect 3 m/cs in a network as in above exercise of PMTU discovery.
2. Send a ping packet A to C with TTL=1.
3. Analyze the response with ICMP errors. Analyze how TTL Expiry works.
4. Analyze the source IP address in the ICMP error packet.

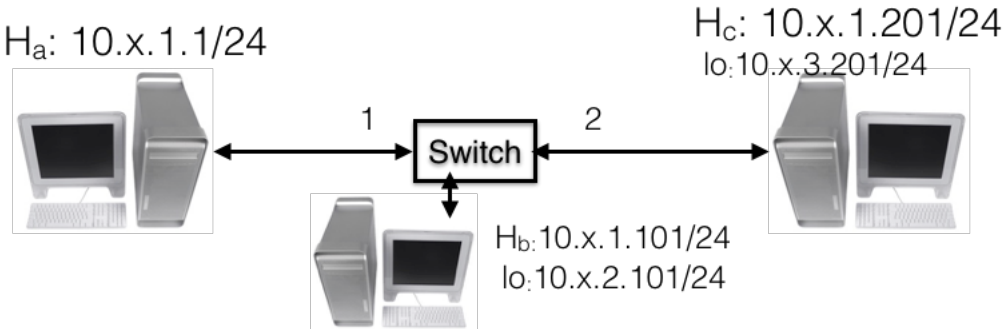
5 Hands-on 3: ICMP Errors

Note:

- The IP addresses specified in the below example are for illustrative purposes only. Use the IP Address assigned to your machines in the network.
- For this exercise you need to make a group of 3.

5.1 ICMP Redirect

1. Connect 4 m/cs in a network as shown in diagram below (via switches). Assign the address manually to these systems. The diagram shows only 3 m/cs, but you can connect the 4th machine accordingly.



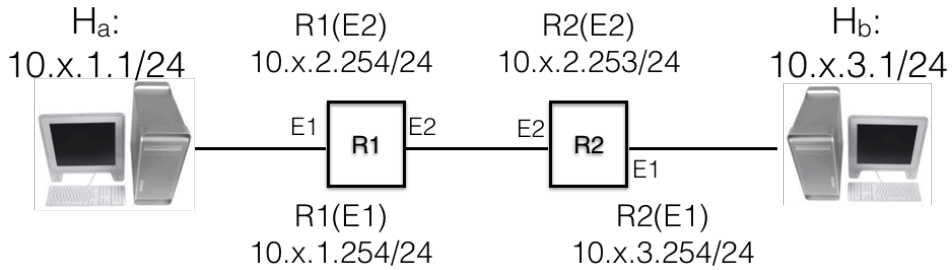
2. On B and C, assign addresses from a different network on the loopback (`lo`) interface?
   a. `sudo ip addr add 10.x.2.101/24 dev lo` on B
   b. `sudo ip addr add 10.x.3.201/24 dev lo` on C
3. Disable acceptance of ICMP redirect on A to ensure that its routing table is not updated as per ICMP redirect. It is assumed that ethernet interface on the machine is `eth0` in the examples below. Replace it with appropriate values.
   a. `sudo sysctl -w net.ipv4.conf.all.accept_redirects=0`
   b. `sudo sysctl -w net.ipv4.conf.default.accept_redirects=0`
   c. `sudo sysctl -w net.ipv4.conf.eth0.accept_redirects=0`

4. Add mis-redirected (incorrect) routing entries on A.
   a. `sudo ip route add 10.x.2.0/24 via 10.x.1.201`
   b. `sudo ip route add 10.x.3.0/24 via 10.x.1.101`

5. Ping from A to addresses `10.x.2.101` and `10.x.3.201`
6. Analyze in `tcpdump` the ICMP redirect packets. Use the capture filter `icmp`.

5.2 PMTU Discovery

1. Connect 4 m/cs in a network as shown in diagram shown below like in section 4.2. Assign the address manually to these systems, and configure the routing entries as defined earlier.



2. Change the MTU value of the link between R1 and R2. Use the interface name appropriately instead of `eno1`.
   a. `sudo ip link set dev eno1 mtu 1000`

3. Send a ping packet bigger than 1000 bytes from A to C. Run following on A
   a. `ping -c1 -s 1200 -p 50515253 10.x.3.201`

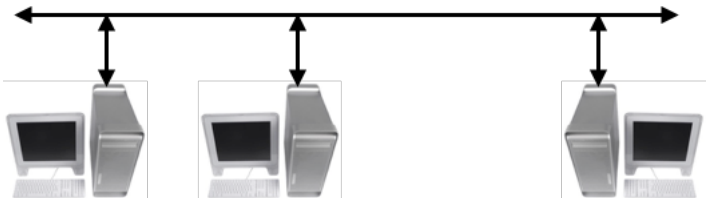
4. Analyze wireshark/tcpdump capture to study ICMP error (fragmentation needed) and subsequent packet fragmentation (This will be studied later). How many packets are finally generated from A. Analyze the ping response. Are there two responses?

5.3 Exercise Expectations

1. Able to understand ICMP errors and source address of packet corresponding to ICMP errors.
2. Able to understand ICMP Redirect, PMTU Discovery and TTL expired.

6 Hands-on 4: Understanding ARP

6.1 Study ARP table maintenance



2. Setup three machines A, B, and C as shown in (in TCP/DUP setup). In this exercise we do not need 2nd and 3rd network i.e,. 10.x.2.0/24 and 10.x.3.0/24 and thus these addresses need not be assigned. Use of separate small switch is not recommended for this exercise
3. Study the existing ARP table and identify which machines are these.
   a. `arp -an`
4. Ping one live machine (i.e. machine which is reachable) and one unreachable machine. Study the ARP table, analyze unreachable machine.
5. Ping the broadcast address of your network. This requires sudo privilege.
   a. `sudo ping -b -c2 10.x.1.255`
6. Analyze in wireshark on number of ICMP response received as well as number of entries added in ARP Table.

6.2 Study Gratuitous ARP

1. Use the same setup as above. Note down MAC Address of 3 m/cs (A, B & C).
2. Install `arping` on the machine A (and may be others)
   a. `sudo apt install arping`
3. On A, add the IP address of A to that of C (replace eth0 with applicable interface name)
   a. `sudo ip addr add 10.x.1.201/24 dev eth0`
4. Issue arping with this new address to B
   a. `sudo arping -s 10.x.1.201 -c 2 10.x.1.101`
5. Study the ARP table at B. It should show updated MAC address of A for IP Address of C.

6.3 Exercise Expectations

1. Able to understand the use of ARP and Gratuitous ARP.

← end of document →