KTH Electrical Engineering

EP2120
Internetworking

Lab 2

Transmission Control Protocol

September 2015

Department of Communication Networks
School of Electrical Engineering
KTH, Royal Institute of Technology
Stockholm, Sweden

Contents

1 Introduction

This syllabus contains information about Lab 2. The topic of Lab 2 is the transmission control protocol (TCP). Please read the syllabus carefully.

1.1 Goals of the Lab

The objective of this lab is the following:

- Demonstrate the difference between UDP and TCP data transmission.
- Show the effect of fragmentation on TCP.
- Give insight into the TCP connection establishment and termination.
- Help understanding of TCP congestion and flow control.
- Expose the difference between TCP bulk and interactive data transfer.

1.2 Requirements and reporting

In order to pass the lab, you need to fulfill the following:

- Answer the preparations found in section 13 before the lab. Hand your solutions to the lab assistant before starting the lab.
- Perform the lab. You need to be present during the lab session.
- Write and submit one lab report per group. The report should be submitted no later than one week after the lab.
- For completing the report you will need to analyze output data and graphs from Wireshark, produced during the lab session.

1.2.1 Lab report

To pass the lab, one lab report shall be written per group. You will probably not have the time to write the report during the lab. Instead, save the logs from `Wireshark` and complete the report after the lab. The lab report shall be written in English and contain the following:

- Names, social security numbers (personnummer) and e-mail addresses of each participant.
- Answers to the questions in the exercises in Sections 5-10.
- `Wireshark` output from the exercises in Sections 5-10, including the summary data, the detailed data, and the graphs. It is explicitly stated in the exercises what data you should save.

The Appendix gives an outline of the lab report.

2 Preparation for the lab

In order to prepare for the lab, read the items in the reading list and answer the questions below. The lab is limited in time, so it is necessary that the preparations are made in advance to the scheduled lab. The schedule is tight.

2.1 Software tools

The following software tools will be used in the lab.

- `ttcp` - Test TCP and UDP performance.
- `tc` - show/manipulate traffic control settings
- `telnet` - An application and a protocol to communicate with another host using TCP.
- `slattach` - Attach a network interface to a serial line using SLIP.
- `ifconfig` - Used to configure a network Interface.
- `route` - Used to manipulate the routing table.
- `import` - Tool to dump a window to a bitmap.
- `wireshark` - An interactive and graphical network traffic analyzer. Read more about Wireshark in Section 2.3.

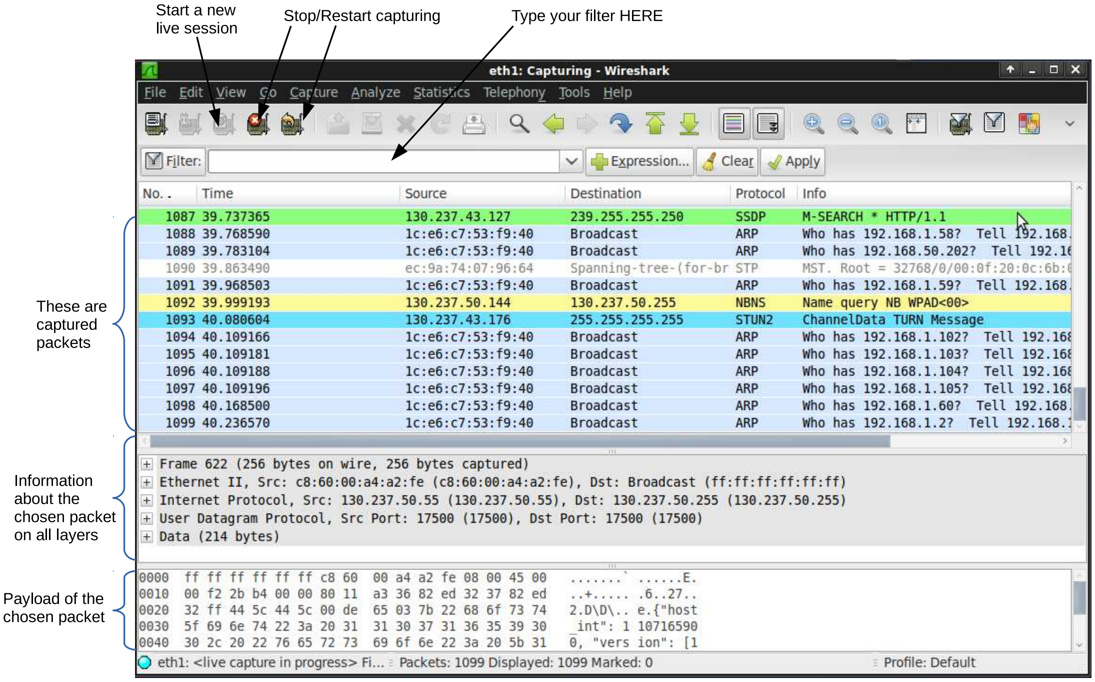
Manual pages of these tools are installed on all Linux computers (man command) and they can also be found on the web (for example http://linux.die.net/). Read the manual pages of these commands. It is even better if you install the above software on your own computer and learn to use them before the lab. More information about Linux traffic control can be found at http://lartc.org/howto.

2.2 Reading list

Forouzan, "TCP/IP Protocol Suite",
4th edition, Chapter 7.3, 8, 11, 12, 13, 14, 15.

- Fragmentation
- UDP
- TCP
- EP2120/IK2218 Lecture Notes: Transmission Control Protocol

Put emphasis on the sections about TCP error and congestion control.


Start a new live session
Stop/Restart capturing
Type your filter HERE
eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply
These are captured packets
Information about the chosen packet on all layers
Payload of the chosen packet

```
No. .  Time        Source               Destination           Protocol Info
  1087 39.737365   130.237.43.127       239.255.255.250       SSDP     M-SEARCH * HTTP/1.1
  1088 39.768590   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.58?  Tell 192.168.
  1089 39.783104   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.50.202?  Tell 192.16
  1090 39.863490   ec:9a:74:07:96:64    Spanning-tree-(for-br STP      MST. Root = 32768/0/00:0f:20:0c:6b:
  1091 39.968503   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.59?  Tell 192.168.
  1092 39.999193   130.237.50.144       130.237.50.255        NBNS     Name query NB WPAD<00>
  1093 40.080604   130.237.43.176       255.255.255.255       STUN2    ChannelData TURN Message
  1094 40.109166   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.102?  Tell 192.168
  1095 40.109181   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.103?  Tell 192.168
  1096 40.109188   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.104?  Tell 192.168
  1097 40.109196   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.105?  Tell 192.168
  1098 40.168500   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.60?  Tell 192.168.
  1099 40.236570   1c:e6:c7:53:f9:40    Broadcast             ARP      Who has 192.168.1.2?  Tell 192.168.1
```

```
+ Frame 622 (256 bytes on wire, 256 bytes captured)
+ Ethernet II, Src: c8:60:00:a4:a2:fe (c8:60:00:a4:a2:fe), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
+ Internet Protocol, Src: 130.237.50.55 (130.237.50.55), Dst: 130.237.50.255 (130.237.50.255)
+ User Datagram Protocol, Src Port: 17500 (17500), Dst Port: 17500 (17500)
+ Data (214 bytes)
```

```
0000  ff ff ff ff ff ff c8 60  00 a4 a2 fe 08 00 45 00   .......` ......E.
0010  00 f2 2b b4 00 00 80 11  a3 36 82 ed 32 37 82 ed   ..+..... .6..27..
0020  32 ff 44 5c 44 5c 00 de  65 03 7b 22 68 6f 73 74   2.D\D\.. e.{"host
0030  5f 69 6e 74 22 3a 20 31  31 30 37 31 36 35 39 30   _int": 1 10716590
0040  30 2c 20 22 76 65 72 73  69 6f 6e 22 3a 20 5b 31   0, "vers ion": [1
```

eth1: <live capture in progress> Fi... Packets: 1099 Displayed: 1099 Marked: 0   Profile: Default

Figure 1: Main `Wireshark` window containing three frames.

2.3 Wireshark

`Wireshark` is a traffic analyzer with a graphical interface. It is the main tool used in this lab. In this section, you find the short description of most of the `Wireshark` functions that are necessary for the lab.

Figure 1 shows the main `Wireshark` window as seen after capturing traffic. The window consists of three frames. A list of the captured packets is shown in the top frame. When a packet in the list is marked, the content is shown in the middle and the lower frames. The middle frame shows the packet in symbolic form, while the lower frame shows the packet in raw, hexadecimal format. In the middle frame, the fields may be expanded to show a more detailed view.

A session is started by choosing *Capture»Start*. This brings up a capture options window. Typically, the display options "Update list of packets in real time" and "Automatic Scrolling in live capture" should be selected. When capturing is started, a capture window is shown and the captured packets are shown in the top frame. To stop capturing, click on the "stop" button in the capture window.

Some statistics can be seen by choosing *Tools»Summary*.

2.3.1 Saving `Wireshark` output

Data can be saved in two formats: summary and detailed. To save data, follow the steps below:

- Choose *File»Print* on the menu bar.
- Choose Plain Text format.
- Choose Print to file.
- Choose a file name (if you have a USB you can save it directly there).
  For saving a summary, choose Print summary.
  For saving detailed data, choose Print details.

2.3.2 Graphs

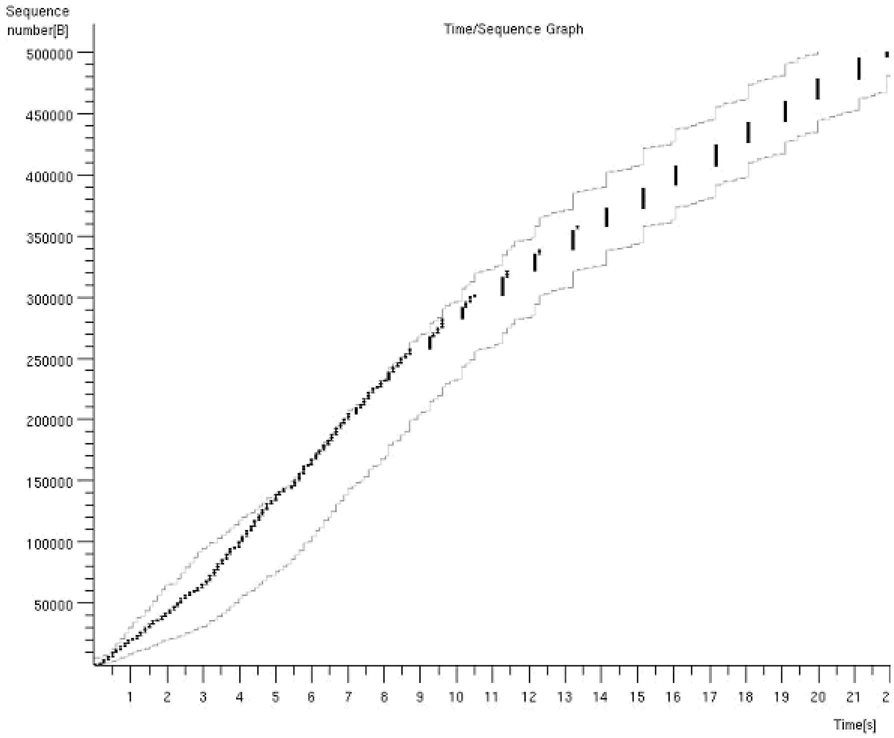


Figure 2: Example of a tcptrace graph.

Figure 2 illustrates a TCP trace graph. After a TCP experiment, this graph is shown by choosing *Statistics» TCP Stream Graph»tcptrace* on the menu bar. The figure shows the time in seconds on the x-axis, and the sequence number on the y-axis. The plots illustrate the segments, and the advertised window.

Unfortunately, `Wireshark` does not allow you to save the graphs to a file. However, by using the import utility, the graph can be dumped to a bitmap. In order to dump the graph to a bitmap do the following. Type

```
# import graph.jpg
```

in a terminal window, then click on the window containing the TCP graph. This saves the graph to a file with the corresponding image format.

`Wireshark` can also be used to create other useful graphs, including the throughput plotted as a function of the time.

3 Lab equipment

Lab groups consist of 2-4 people. Every group will be provided with the following equipment:

1. Two Microstar routers with four Ethernet interfaces and one serial interface each running Linux (RedHat 9).

2. One cross-wired serial cable (null-modem).

3. Three crossed Ethernet cables.

4. One straight Ethernet cable. (Used to connect to the Internet at the end of the lab.)

Every group will also need two end-hosts equipped with one Ethernet interface each, running Linux and the software listed in Section 2.1. The choice of computers that you will use remains the same as in the first lab exercise: you can either bring your own computer or use one of the laboratory laptops.

4 Lab setup

On a commencé par se connecter au router via ssh puis on a up son interface eth0 avec son @IP puis on a configuré notre host avec son @IP sur eth1 puis on s'est reconnecté au router via ssh et on fait les routes puis on a autorisé le fowarding de paquet pour pouvoir ping du host A au host B donc tout marche dorénavant.

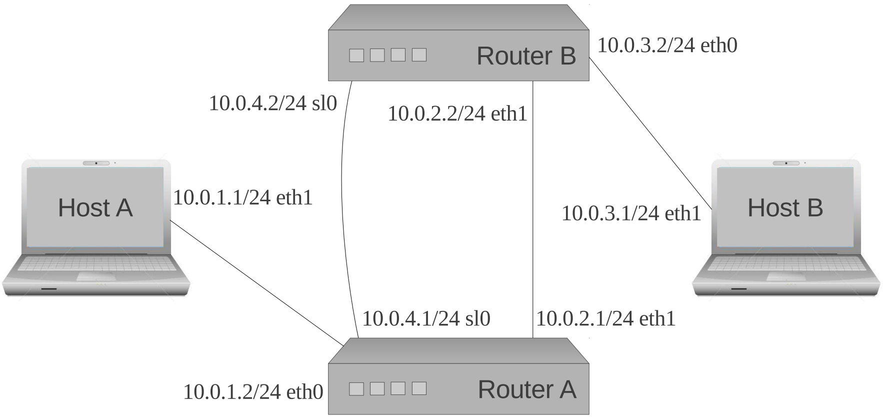


Figure 3: Two hosts, A and B, interconnected by routers A and B. Router A and Router B are connected by an Ethernet cable and a serial cable (cross-connect)

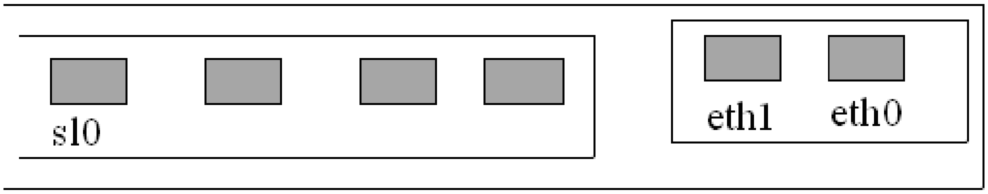


Figure 4: Illustration of the port numbering on a router. This is the rear view of the router.

4.1 Connecting the network

In the lab set up, hosts A and B are connected to routers A and B respectively, according to Figure 3. The two routers are connected with one Ethernet cable and with one serial cable. Use the interfaces as specified in Figure 3. The interface numbering of the routers is depicted in Figure 4.

4.2 Configuring the hosts

Turn on the hosts. The hosts will boot Linux and start a window system. Once the system boots you will be automatically logged in as:

```
Username:  user
password:  1234
```

As stated in the manual of Lab 1, in the Ubuntu distributions the user is strongly encouraged to login and work as a simple user and use `root` privileges only when necessary via the `sudo` command. However, in this lab since we are going to mainly use system commands, the constant use of `sudo` might become annoying. Therefore, during this lab session, starting a shell as `root` is recommended. You can do so by typing

```
    sudo -s
```

Open up a terminal window (right click on the desktop and choose "New Terminal"). To make it easy to remember which host you are working on, set the prompt to reflect the name of the host according to Figure 3.

```
root@live:∼# cat » /root/.bashrc
export PS1=’’Host A# ‘‘
ctrl-D
root@live:∼# source /root/.bashrc
Host A#
```

The next step is to configure the Ethernet interfaces. The routers have the IP address 10.0.0.1/24 pre-configured on eth0. Set the IP address of the Ethernet interface of the host so that you can communicate with the router via `ssh`.

The last step is to ensure that the host will not make DNS lookups. A straightforward way to do this is to remove the /etc/resolv.conf file.

Repeat the host configuration procedure for Host B.

4.3 Configuring the routers

The Ethernet interfaces for router A and router B are already configured. After connecting the cables you should be able to login to router A from host A. Start a new terminal window and use `ssh` to connect to the router. Remove the /root/.ssh/known_hosts file first if it exists.

```
Host A# rm /root/.ssh/known_hosts
Host A# ssh 10.0.0.1
Router A# password for root:  qwerty
```

Log in to router B from host B in the same way.

After the login, you should change the IP address of the eth0 interface of the routers according to Figure 3. Note that after changing the IP addresses of the routers' eth0 interfaces you must change the IP addresses of your hosts, so that they match the new networks.

Add the router's new IP addresses as the default gateway on the hosts.

Use `ssh` again to connect to the routers.

In the routers, change directory to /home/ep2120_lab2. Create the directory if it does not exist.

Add the routes to the 10.0.3.0/24 network for router A and the routes to the 10.0.1.0/24 network for router B.

Make the router start forwarding traffic by writing the following line at the routers:

```
Router A# echo "1" > /proc/sys/net/ipv4/ip_forward
Router B# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Verify that the setup is correct by pinging Host B from Host A.

After you verified the connectivity, attach the serial line and assign IP addresses to the serial interfaces. The name of the serial line interface is sl0.

```
Router A# slattach -p slip -s 4800 /dev/ttyS0 &
Router B# slattach -p slip -s 4800 /dev/ttyS0 &
Router A# ifconfig sl0 10.0.4.1 pointopoint 10.0.4.2 up mtu 600
Router B# ifconfig sl0 10.0.4.2 pointopoint 10.0.4.1 up mtu 600
```

Verify that you can ping from 10.0.4.1 on router A to 10.0.4.2 on router B.

At this point, the traffic between the two hosts goes over the fast Ethernet link. You can make the traffic go through the serial line by updating the routes on Router A and Router B respectively. Do not change the routes at this step.

The `arp` tables at the hosts and the routers will timeout every minute, this may affect your measurements. So for all the routers and all the hosts, do the following:

```
# echo 30000 > /proc/sys/net/ipv4/neigh/default/gc_interval
```

You have now successfully set up the measurement environment.

5 Measuring TCP with `ttcp` and `Wireshark`

This exercise is an introduction to measuring TCP traffic. You will use **ttcp** to send TCP traffic from host A to host B, and you will use **Wireshark** to capture the traffic on Host B. This exercise is very similar to the exercise that you performed during Lab 1 for UDP. Note however that **ttcp** does not send any extra packets when running TCP.

1. On Host B, start **Wireshark** to measure traffic on interface eth0, add a filter so it will only measure the traffic from and to host 10.0.1.1. Done

2. On Host B, start capturing packets with **Wireshark**. (Choose *Capture»Start* on the menu bar to start capturing the traffic (see Section 2.3).) Done

3. On Host B, start a **ttcp** receiver to receive TCP traffic at port number 1234.
   ttcp -rs -p1234

4. On Host A, start a **ttcp** sender to send 10 packets with the length of 1000 bytes using TCP to Host B, at port number 1234. ttcp -ts -p1234 -l1000 -n4 10.0.3.1

5. When the transfer is completed, stop capturing. You should now have a packet trace in the main window. Make sure that you recognize the Ethernet header fields, the IP header fields, and the TCP header fields in the mid section.

6. Save the output (detailed and summary) to file output_5_detail and output_5_summary. (Section 2.3 shows in detail how to do it.)

7. Observe the traffic and answer the following questions:

   (a) How many packets are transmitted in total (count both directions)? 20 counting the 3 way handshake !

   (b) What is the range of the sequence numbers used by the sender (Host A)?
   Sent SEQ from 1 to 1002 (end of the discussion)

   (c) How many packets do not carry a data payload?
   The 3 last one (end of discussion) and every ack and the 3 way handshake so 12

   (d) What is the total number of bytes transmitted in the recorded transfer? (Calculate the amount of user data that was transmitted!) we look how many datas are sent from the 8 other packets, and we do total - this
   total : 11336 bytes
   user = total - 10 000
   = 1336
   So perfect we sent 10000 bytes !

   (e) Compare the total amount of data transmitted in the TCP data transfer to that of a UDP data transfer. Which of the protocols is more efficient in terms of overhead? What is the efficiency in percentage for these two protocols? (Recall the UDP measurements from the previous lab. How many bytes were sent in total using UDP?) Check with the last lab and do the question

Lab report: Use the captured data to answer the questions above. Support your answers with the saved **Wireshark** data. Append the file output_5_summary to the report.

6 TCP connection management

This exercise gives an insight in TCP connection establishment and termination. In this exercise we use the **telnet** application to establish and terminate TCP connections.

6.1 Connection establishment and termination

1. Start capturing packets on Host B with Wireshark.

Host A :
telnet 10.0.3.1

Host B :
/

2. Establish a telnet connection from Host A to Host B.

3. On Host A, terminate the connection. Type ctrl-] (ctrl+AltGr+9) at the telnet prompt and then type ‘‘quit’’.

4. Stop the Wireshark capturing.

5. Save a summary Wireshark output to the file output_6_1_summary.

6. Study the list of captured packets. Observe the TCP connection establishment and answer the following questions:

   (a) Which packets constitute the three-way handshake? Which flags are set in the headers of these packets?

   The 3 first one, TCP packets. First packet : flags SYN : set / Second packet : flags SYN, ACK : set / Third packet : flags ACK So we have the 3 way hand shake proper to TCP

   (b) What are the initial sequence numbers used by the client and the server, respectively? Server (Host B) : Sequence number 1 and same for the client (Host A).

   (c) Which packet contains the first application data? The fourth packet directly after the 3 way handshake

   Initial window sizes
   client : 5888
   server : 5792

   (d) What are the initial window sizes for the client and for the server?

   (e) How long does it roughly take to open the TCP connection?

   0.001097 s so 1ms rougjly (we just do time ACK less time SYN)

7. Study the list of captured packets. Observe the TCP connection termination and answer the following questions.

   (a) Which packets are involved in closing the connection?

   It’s a TCP connection procedure.

   (b) Which flags are set in these packets? The flags FIN,ACK then FIN,ACK and the last one : ACK.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved Wireshark data. Append the file output_6_1_summary.

6.2 Connecting to a non-existing port

In the following exercise you will see what happens when you try to establish a TCP connection to a non-existing port. This could be the case when you try to load a home page from a host that does not have a web server.

1. Start capturing packets with Wireshark on Host B.

2. Try to make a telnet connection from Host A to a port on Host B without listeners.

3. Stop capturing packets.

4. Save a summary Wireshark output to file output_6_2_summary.

5. Study the captured list of packets and observe the TCP segments that are transmitted. Answer the following questions.

   (a) How does the server host (Host B) close the connection? By sending RST flags (reset flags)

   (b) How long does the process of ending the connection take? 45 microseconds

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_6_2_summary.

6.3 Connecting to a non-existing host

In the following exercise you will see what happens when you try to establish a TCP connection to a non-existing host. This could be the case when you try to load a home page from a host given with its IP address that does not exist.

1. On Host A, start capturing packets with `Wireshark`.

2. Set a static `ARP` entry to a non-existent host. Without this fix, no TCP packets will be sent. Instead, Host A would start sending ARP requests and receive no answers.
   ```
   Host A# arp -s 10.0.1.42 1:2:3:4:5:6
   ```

3. Try to make a `telnet` connection to host with the above IP address from host A.

4. Wait for up to 5 minutes and then stop capturing packets and save the `Wireshark` output to file output_6_3_summary. We need to wait 5 minutes, trololololol

5. Study the captured packets and observe the TCP segments that are transmitted. Answer the following questions.

   (a) How often does the client try to open a connection? Note the time interval between attempts. 3 times within 5 minutes, there is 3 seconds between the first and the second connection and 6 seconds between the second and the third one.

   (b) Does the client stop trying to connect at some point? If so, after how many attempts? Yes after 3 attempts he stops trying.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_6_3_summary.

7 Fragmentation in TCP

In this exercise you will observe the effects of fragmentation in TCP. Note that it is common to make a mistake in this exercise. Please show the output you get to a lab assistant in order to ensure that you have the correct results.

1. Start capturing packets with `Wireshark` on Host B. For this exercise it is convenient to set the Display options "Update list of packets in real time" and "Automatic scrolling in live capture".

Important to set the MTU to 600
IN THE ROUTER - - ‘

2. Start Wireshark on Host A as well
   ```
   Host A# Wireshark -ni eth0 -f "host 10.0.3.1 or icmp" &
   ```

3. Change the MTU of eth1 on Router A to 600.

4. Repeat the ttcp measurement from before.
   ```
   Host B# ttcp -rs -p1234
   Host A# ttcp -ts -l1000 -n10 -p 1234 10.0.3.1
   ```

5. When the transfer is completed stop capturing packets on Host A and Host B.

6. Observe the traffic and answer the following questions.

   (a) How many packets did Host A measure and how many packets did Host B measure? Why?
   (Notice that there are differences between the two Wireshark measurements!)

   (b) Is the DF flag set in the datagrams? Why? It is set in the IP protocol. Maybe because they are divided by the TCP protocol and not by the IP protocol (?).

   (c) Do you observe fragmentation? If so, where does it occur? No.

   (d) Study the ICMP messages recorded at Host A.
   Which node is the source? The source is the router
   What is the type and the code of the messages? Destination unreachable (fragmentation needed)

7. Reset the MTU of eth1 on router A to 1500.

8. You must also update Host A to use the larger MTU again
   ```
   Host A# route del default
   Host A# route add default gw 10.0.1.2
   ```

9. Save the Wireshark output to files output_7_A_summary, output_7_A_detail, and output_7_B_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved Wireshark data. Append the file output_8_A_summary and output_8_B_summary to the report. Append also the part of output_8_A_detail that shows one of the ICMP messages. (You can edit the file in a text editor to cut out the required information.) A : 47 // B : 33 - Because MTU was too small on the router, so we sent to big packets and so the router sent information back. It comes from the negociation between the host and the router about the MTU

8 TCP data transfer

In this exercise you will study TCP flow control and some properties of TCP data transfer. TCP consists of several heuristics to cope with different network and traffic conditions. In particular, TCP has different behaviour for interactive applications and for bulk transfer. TCP also has different behaviour on a fast link and on a slow link.

The interactive application in this exercise is telnet, the bulk data transfer is done with ttcp. The fast link is an Ethernet link, while the slow link is a SLIP serial link (see Table 8).

	Interactive application	Bulk data transfer
Fast link	Telnet over Ethernet	Ttcp over Ethernet
Slow link	Telnet over Serial	Ttcp over Serial

Table 1: TCP data transfer classification.

8.1 Interactive application - fast link

1. Establish a `telnet` connection from Host A to Host B on the fast link. Log in with as user *user* with the credentials mentioned above.

2. Start capturing packets with `Wireshark` on Host A.

3. Type a few characters in the `telnet` application. The `telnet` client (Host A) sends each character in a separate TCP segment to the server (Host B) which in turn echoes the character back to the client. Including echoes, we would therefore expect to see four TCP segments for each typed character.

   (a) How many segments can be seen? Only 3 including 2 Telnet Data

   (b) Describe the payload of each packet. First segment : Telnet Data : data : a //Second segment : telnet data : data : a // third segment : TCP : no payload just a ack

   (c) Explain why you do not see four packets per typed character. Because the acknowledgment from the server is piggybacked with the telnet data.

   (d) When the client receives the echo, it waits a certain time before sending the ACK. Why? How long is the delay? Because it may used delayed ack. ~0.025 s : 25 ms

   (e) In the segments that carry characters, what window size is advertised by the `telnet` client and by the server? Does the window size vary as the connection proceeds? Sender : 312 Receiver : 181 No.

4. Type quickly a lot of characters in the `telnet` application, such as by pressing a key continuously.

   (a) Do you observe a difference in the transmission of segment payloads and ACKs? We send the ack only after the echo ack returns. We do have a piggybacked ack we can see it on that capture.

5. Stop the capturing of packets.

6. Save the `Wireshark` output to file output_8_1_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_8_1_summary to the report.

8.2 Bulk transfer - fast link

In this exercise, the behavior of TCP is examined when large amounts of data are transmitted. TCP uses the acknowledgements to limit the sending rate; this together with the receiver window size is the basis of flow control.

1. Start capturing packets with Wireshark on Host A.
2. Start a ttcp receiver on B and a sender on A, send 1000 packets of length 1000 bytes each.
3. Stop capturing packets.
4. Draw a tcptrace graph. Study the graph carefully.
5. Observe the sliding window protocol from the output of Wireshark. The sender transmits data up to the window size of the receiver. Answer the following questions.
   (a) How often does the receiver send ACKs? Can you see a rule on how TCP sends ACKs? We send an ACK for every 2 segments received. Delayed ACK ?
   (b) How many bytes of data does a receiver acknowledge in a typical ACK? 2 segments so ~2000 bytes.
   (c) How does the window size vary during the session? For the sender it never change 5888 bytes all the time
   (d) Select any ACK packet in the Wireshark trace and note its acknowledgement number. Find the original segment in the Wireshark output. How long did it take from the transmission until it was ACKed?
   (e) Does the TCP sender generally transmit the maximum number of bytes as allowed by the receiver?
6. Save the tcptrace graph to file output_8_2_tcptrace.jpg and save the Wireshark packet trace to file output_8_2_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved Wireshark data. Append the output_8_2_tcptrace.jpg to the report.

8.3 Interactive application - slow link

1. On router A and router B, change the routes so the traffic between the hosts will go through the slow link.
2. Establish a telnet connection from Host A to Host B on the slow link Log in with *qwerty* as the root password.
3. Start capturing packets with Wireshark on Host A.
4. Type a few characters in the telnet application. Vary the rate at which you type characters. Observe the output from Wireshark. Answer the following questions.
   (a) How many packets are transferred for each keystroke? Does the number change when you type faster?
   (b) Do you observe delayed acknowledgements?
   (c) Do you observe the effect of Nagle's algorithm? How many characters can you see in a segment?

5. Stop capturing packets.

6. Save the Wireshark output to file output_8_3_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved Wireshark data. Append the file output_8_3_summary to the report.

8.4 Bulk transfer - slow link

1. Start capturing packets with Wireshark on Host A.

2. Start a ttcp receiver on host B and a sender on host A. Send 50 packets of the length 1000 bytes each.

3. Stop capturing packets.

4. Draw a tcptrace graph. Study the graph carefully.

5. Observe the differences compared to the bulk data transfer on the fast link. Answer the following questions.

   (a) Look at the pattern of segments and ACKs. Did the frequency of ACKs change compared to the bulk transfer on the fast link? How?

   (b) Are the window sizes advertised by the receiver different from those of the previous exercise?

   (c) Does the TCP sender generally transmit the maximum number of bytes as allowed by the receiver?

6. Save the tcptrace graph to file output_8_4_tcptrace.jpg and save the Wireshark packet trace to file output_8_4_summary.

Lab report: Use the captured data to answer the questions above, with special emphasis on the difference with the bulk transfer on the fast link. Support your answers with the saved Wireshark data. Append the output_8_4_tcptrace.jpg to the report.

9 TCP retransmissions

In this exercise you will study TCP retransmissions. TCP uses ACKs and timers to trigger retransmissions of lost segments. In order to cause retransmissions, you will have to artificially introduce errors on some of the links. To do so, you are going to use tc, a kernel module for manipulating traffic control settings. In this exercise you are going to introduce losses on the interface *eth1* of host *B*. The command to introduce losses is:

```
tc qdisc add dev eth1root netem loss 1%
```

The above command tells the kernel to introduce a 1% loss at the interface *eth1*, namely to drop on average 1 every 100 packets going out from that interface. To restore zero losses you can simply type

```
tc qdisc change dev eth1root netem loss 0%
```

or you can cancel outright the queueing discipline between the network layer and the network adapter if you are not going to use it later on. Cancelling the queueing discipline is done by invoking the command

```
tc qdisc del dev eth1root
```

1. Start a second terminal window (Terminal 2) as `root` with `sudo -s`. You are going to use this terminal to switch on and then off losses at step 4

2. Start capturing packets with `Wireshark` on Host A.

3. Start a `ttcp` receiver on host B and a sender on host A, send 50 packets of length 1000 bytes each.

4. When at least 40 packets have been captured by `Wireshark`, in Terminal 2 introduce 100% losses at the interface *eth1* of Host *B*. Wait for one minute (watch `Wireshark` and wait for a few packets to be transmitted) then reinstate zero losses at *eth1*.

5. When the transfer is completed, stop capturing packets.

6. Study the list of captured packets. Draw a tcptrace graph. Observe when retransmissions take place (during the "disconnection" of the Ethernet cable) and answer the following questions.

   (a) How many packets are transmitted at retransmission timeout?

   (b) Do the retransmissions end at some point?

7. Save a summary `Wireshark` output to file output_9_1_summary.

8. Save the tcptrace graph to file output_9_1_tcptrace.jpg.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the `Wireshark` data (output_9_1_summary) and the tcptrace graph (output_9_1_tcptrace.jpg).

10 TCP congestion control

In this exercise, the behaviour of TCP congestion control is examined. TCP congestion control operates in two phases, called slow start and congestion avoidance. Congestion is simulated by introducing losses on the link between Host *A* and Router *A*.

1. Configure the routes so the traffic from Host A to Host B goes through the fast link and the traffic from Host B to Host A goes through the slow link.

2. Start a new terminal on Host A and introduce a 1% loss rate for the outgoing traffic at interface *eth1* using the `tc` command as in the previous exercise.

3. Start capturing packets with `Wireshark` on Host A.

4. Start a `ttcp` receiver on Host B and a sender on Host A, send 2000 packets with a length of 1000 bytes.

5. After the transmission is completed go carefully through the list of captured packets by Wireshark. What do you observe?

6. Draw a tcptrace graph in `Wireshark`.

   (a) Try to observe periods when TCP sender is in slow start phase and when the sender switches to congestion control. Verify if the congestion window follows the rule of the slow-start phase.

   (b) Can you find occurrences of fast recovery?

7. Save a summary `Wireshark` output to file output_10_1_summary.

8. Save the tcptrace graph to file output_10_1_tcptrace.jpg.

9. Configure the routes so the traffic from Host B to Host A goes through the fast link again. Go through steps 3 - 6 again. Include `Wireshark` data and the tcptrace graph to support your answer. Annotate the events in the tcptrace graph, and explain the events that you observe.

11 Link Sharing

The `qdisc` scheduling we used previously does not distinguish between different classes of traffic. In this exercise we will use priority queuing to share the bandwidth of a link among competing flows and we will see how TCP reacts. Such a solution can be used, for example, when you have one physical link and you want to divide it into several slower links for different purposes (Figure 5).

Recall that priority queuing is used in the Differentiated Services (DiffServ) architecture, and the priority of a datagram is determined by its ToS field. Unlike in DiffServ, in this exercise we do not use the ToS field in the IP header to assign priority to datagrams but we create classes of flows based on the TCP port that they use.

To divide the link capacity, we use a packet scheduler called *Hierarchical Token Bucket* (HTB). HTB assigns capacity to classes of flows, and ensures that the amount of service provided to each class is at least the minimum of the amount it requests and the amount reserved for it.

First, we configure the HTB queuing discipline at the interface *eth1* and give it the "handle" 1. We then limit the rate of the outgoing link to 12*kbps*.

```
tc qdisc add dev eth1 root handle 1:0 htb default 3
tc class add dev eth1 parent 1:  classid 1:1 htb rate 12kbps ceil 12kbps
```

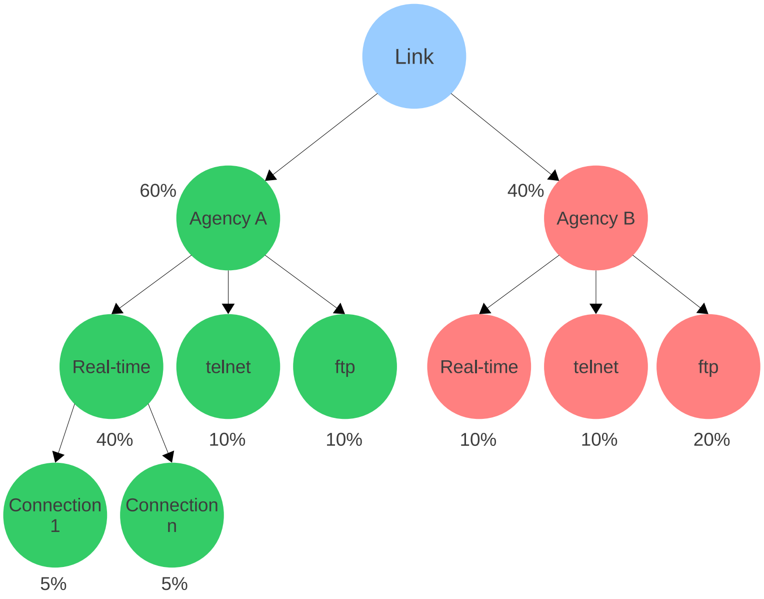


Figure 5: A hierarchical link-sharing structure

Next, we divide traffic into two classes of flows, both have an average (assured) rate of 5$kbps$ and a maximum (ceil) rate of 12$kbps$.

```
tc class add dev eth1 parent 1:1 classid 1:2 htb rate 5kpbs ceil 12kbps
tc class add dev eth1 parent 1:1 classid 1:3 htb rate 5kpbs ceil 12kbps
```

We create filters for the two classes of flows and we assign priority 0 to the high-priority traffic and priority 1 to the low-priority traffic.

```
tc filter add dev eth1 protocol ip parent 1:0 prio 0 u32 match ip
dport <dest_port_1> 0xffff flowid 1:2
tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip
dport <dest_port_2> 0xffff flowid 1:3
```

Traffic that doesn't match any of the filters will be given an assured rate of 2kbs (i.e., the remaining bandwidth).

```
tc class add dev eth1 parent 1:1 classid 1:4 htb rate 2kpbs ceil 12kbps
```

1. On Host B, start capturing packets with `Wireshark` and set `ttcp` listeners on two ports.
2. On Host A, start sending 200 low-priority packets with a length of 1000 bytes. The destination port should be $dest\_port\_2$ to match the configured filter.
3. When around 50 packets are sent, establish the high-priority flow by sending 100 packets with a length of 1000 bytes to destination port $dest\_port\_1$.
4. After the transmission is finished, observe the list of the captured packets.
5. What happened when you introduced the high-priority flow?
6. Calculate the rates at which the packets were sent.

12 Completing the lab

In order to complete the lab, you will have to put together all the output of Wireshark (packet data and graphs). If you used your USB stick then you are done, if you saved your data in the routers then you will have to transfer it to one place.

1. Collect the output into one common archive. One way to do this is to copy all files over to Host A, and then build an archive.
   ```
   Host A# scp IP_of_Host_A:/home/user/* .
   Host A# tar czf lab2.tgz *
   ```

2. If your output is not directly saved on a USB stick you can transfer the archive to your personal mailbox or computer by connecting to the Internet. The lab assistants will give you instructions on how to connect to the Internet and upload your archive.

3. Transfer the archive to a remote Internet site by using ftp, scp, mozilla, or some other tool.

4. Erase the files from the routers and power them down.
   ```
   Router A# rm -rf /home/ep2120_lab2
   Router B# rm -rf /home/ep2120_lab2
   Router A# halt -p
   Router B# halt -p
   ```

5. Power down the hosts.
   ```
   Host # halt -p
   ```

6. Disconnect the cables and ensure that the equipment is in the same state (or better) as when you started.

EP2120 - Internetworking
Lab 2 - Entry test questions

Student name: ____

The answers to the questions should be handed in at the beginning of the Lab session. The manual pages of the commands listed in Section 2.1 and the material listed in the Section 2.2 can help you in answering these questions.

1. Explain the role of the port numbers in the transport layer.

Because we need second identifiers to define the processes

2. Write the command to configure a network interface eth0 to 10.0.0.1 with subnet mask 255.255.255.0 and with an MTU of 600.

ifconfig eth0 10.0.0.1 netmask 255.255.255.0 mtu 600

3. Write the commands to add/delete 10.0.1.2 as your default gateway.

route add/del default gw 10.0.1.2 eth0

4. Write the commands to add/delete 10.0.2.2 as your route to the 10.0.3.0/24 network.

route add/del 10.0.3.0/24 10.0.2.2

5. Write the ttcp commands for both sender and receiver, which executes the following scenario: Send a TCP stream from host 10.2.3.4 to 10.4.5.6 on port 3333. The sender should send 4000 bytes with four datagrams of 1000 bytes in each datagram.

ttcp -t -u -l -1000 -n 4 s -p 3333 // ttcp -r -u -p 3333

6. Write the command to start an `Wireshark` session that captures packets on Ethernet interface eth0. The session should not do DNS name lookup, and should only capture traffic with destination IP address 10.0.0.1. It should also make live capturing and update the display as packets are captured in real-time.

wireshark -i eth0 -f ip.addr==10.0.0.1 -N C

7. Which fields in the IP header are related to fragmentation and what role does each of these fields have?

The Identification field, and Fragment offset field
along with Don't Fragment and More Fragment
flags

8. Suppose a TCP sender receives an ACK packet in which the acknowledgement number is set to 12345 and the window size is 2048. Which sequence numbers can the sender transmit?

SEQ 14393

9. Briefly describe the following algorithms and when they are used:

   (a) Nagle's algorithm.

   (b) Karn's algorithm.

   (c) Delayed acknowledgement.

   (d) Piggybacked acknowledgement.

10. How is the retransmission timeout (RTO) value computed in TCP?

11. Explain the following TCP mechanisms:

   (a) Sliding window flow control.

   (b) Slow start and congestion avoidance.

   (c) Fast retransmit and fast recovery.

To compute the current RTO, a TCP sender maintains two state
variables, SRTT (smoothed round-trip time) and RTTVAR (round-trip
time variation).
Until a round-trip time (RTT) measurement has been made for a
segment sent between the sender and receiver, the sender SHOULD
set RTO <- 3 seconds
When the first RTT measurement R is made, the host MUST set

SRTT <- R
RTTVAR <- R/2
RTO <- SRTT + max (G, K*RTTVAR)

where K = 4.
When a subsequent RTT measurement R' is made, a host MUST set

EP2120
Internetworking
Lab 2
Transmission Control Protocol

Preparation Questions
Measurements

- Measuring TCP with `ttcp` and `Wireshark`
- TCP connection management
  - Connection establishment and termination
  - Connecting to a non-existent port
  - Connecting to a non-existing host
- Fragmentation in TCP
- TCP data transfer
  - Interactive application - fast link
  - Bulk transfer - fast link
  - Interactive application - slow link
  - Bulk transfer - slow link
- TCP retransmission
- TCP congestion control
- Appendix: `Wireshark` output (Please just include interesting parts from the files)
  - output_5_summary
  - output_6_1_summary
  - output_6_2_summary
  - output_6_3_summary
  - output_7_A_summary
  - output_7_B_summary
  - output_7_A_detail (subset)
  - output_8_1_summary
  - output_8_2_tcptrace.jpg
  - output_8_3_summary
  - output_8_4_tcptrace.jpg
  - output_9_1_summary

  - output_9_1_tcptrace.jpg
  - output_10_1_summary
  - output_10_1_tcptrace.jpg