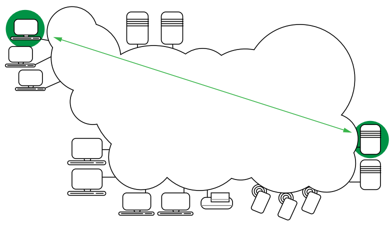
ComNet - Lab n°4

Transport Layer (1): TCP and UDP



1 Review of the transport layer

1. A web client wishes to access a document for which it knows the URL. The IP address for the server is initially unknown. Which application layer protocols are required to satisfy this request?
2. Which transport layer protocols are needed to satisfy the same request?

1.1 Connectionless protocol, UDP

1. Review the characteristics of a connectionless protocol.
2. Indicate the main features of UDP.
3. Explain why a developer would choose UDP over another transport protocol.
4. In your opinion, can a UDP-based application transmit data reliably? Justify your response.

1.2 Connection oriented protocol, TCP

1. Review the characteristics of a connection oriented protocol.
2. Describe the mechanisms that are necessary in order to ensure reliable data transfer.
3. Indicate the main features of TCP.

1.2.1 Connection management

1. How is a segment's role indicated in TCP?

2. Draw the diagram to establish a network connection. Discuss the number of messages required. In the context of TCP, why proceed a three phases exchange?

3. What are the possibilities for numbering of the segments? In the context of TCP, how the sequence numbers evolve? Could two successive segments hold the same sequence number? And in the absence of data transmitted, may the sequence number increase?

4. Why not start the numbering of sequence to 0?

5. What is the potential termination of a connection? Sketch the corresponding diagrams.

1.2.2 Reliability management

1. The reliability requires knowledge of the reception of data. What are the two main techniques to perform the acknowledgment? Precise their respective interests according to the traffic control and which is used by TCP

2. In case of data loss, two retransmission policies are possible: Describe them and precise the one used with TCP.

1.2.3 Estimated RTT of a connection

When using TCP, the choice of the RTT (Round Trip Time) is important since the detection of loss depend and various sending control mechanisms will directly depend of this parameter. The calculation of RTT can result of the following formula $RTT = \alpha * RTT_{mesure} + (1 - \alpha) * RTT_{old}$ with α the smoothing factor.

1. How TCM measure the round trip delay (RTT_{mesure}) for a given segment?

2. Show that the effect of a measured value for the RTT is reduced exponentially with time.

3. What is the interest to use this formula comparatively to a "sliding average" in which the RTT is the average taken on a window of length L?

4. What are the consequences of a value of α close to 1 or close to 0?

5. What precautions to take when measuring the round trip delay of a given segment?

6. In your opinion, what is the usefulness of the TCP `timestamp` option? Why is it advisable to use this option (RFC 1323 can be see for details)?

1.2.4 Calculation of TCP RTO

1. The first approach to determine the value of the retransmission timer RTO (Retransmission TimeOut) is $RTO = n*RTT$. What precautions to take regarding the size of n?

2. The second approach uses $RTO = RTT + \delta D$ usually with $\delta = 4$.
   $D = \beta(|RTT_{mesure} - RTT_{old}|) + (1-\beta)D_{old}$ generally with $\beta = 1/4$.
   This approach consists in computing the variance of RTT. What is the improvement?

3. How to calculate the RTO when there are losses?

2 Observation of UDP traffic

The following analyzes are intended to observe the mechanisms of UDP protocol.

2.1 Characteristics of a UDP datagram (without computer assistance)

Here is the trace of a frame to study:

```
0000  00 04 76 21 1b 95 00 01  02 a5 fb 88 08 00 45 00   ..v!.... ......E.

0010  00 30 00 00 40 00 40 11  6d 58 c2 fe a3 b1 c2 fe   .0..@.@. mX......

0020  a3 b6 06 9c 00 45 00 1c  e1 e4 00 01 75 6e 69 78   .....E.. ....unix

0030  62 6f 74 74 00 6e 65 74  61 73 63 69 69 00         bott.net ascii.
```

1. Manually analyze (without `wireshark`/`tshark`) the frame presented above. Use directy the lab support to $\boxed{\text{surround}}$ the various fields on the trace.
2. Which information can we deduce from the ports number contained in the above datagram
3. UDP est dit minimaliste en terme de fonctionnalités. En observant les champs présents dans l'en-tête, pensez-vous que leur nombre soit réduit au maximum ?

2.2 Capture and analysis of UDP datagrams

2.2.1 Capturing UDP traffic

First, you will capture some UDP traffic in order to understand its basic characteristics. Topology 1 (with client and server on the same LAN, as described in Lab n°1) is available to you on the networking testbed. Generate some UDP traffic with TFTP. Capture this traffic using `wireshark` or `tshark`, as follows:

- From PPTI PC **N**, connect to the 3 corresponding VMs of the testbed from 3 separate terminal windows
  - access in the "client" vm**N**1 (window 1) with SSH to `etudiant@10.0.7.`**N**`1`
  - access in the "monitor" vm**N**2 (window 2) with SSH to `etudiant@10.0.7.`**N**`2` (use `-Y` if you want to run `wireshark`)
  - access in the "server" vm**N**3 (window 3) with SSH to `etudiant@10.0.7.`**N**`3`
- Verify that the TFTP server is running on `10.0.7.`**N**`3` (window 3)
  - look for the server process (type: `ps aux | grep tftp` or have a look in `inetd.conf`)
  - verify the TFTP server directory is configured to realize the transfers
  - look at the interfaces and especially the one of the experimental LAN (`/sbin/ifconfig eth1`) and verify the IPv4 server address for the client connection (should be `10.`**N**`.1.`**N**`3`)
- Start the capture by running the sniffer on `10.0.7.`**N**`2` (window 2)
  - run the sniffer by typing: `wireshark`
  - initiate the capture on interface `eth1`, as described in **Lab n°1**
- Start a TFTP client on the "client" VM (window 1)
  - type `tftp 10.`**N**`.1.`**N**`3`, which should make a TFTP exchange from the "client" VM to the "server" VM

  - get a file from the server with the `get` command
  - complete the exchange with the `quit` command

- Observe the trace captured in the `wireshark` window
- **Filter the traffic to keep only UDP** (filter = `udp`). Save the filtered trace for later reuse. Keep the application running in order to conduct the following analysis.

2.2.2 Analysing the UDP exchange

1. Compare with the capture of similar TFTP exchanges of **Lab n°2** (with filter = tftp), what differences do you notice with the trace **displayed** here?

2. Can you identify the roles of implicated applications client or server ?

3. How is managed the association of the two implicated applications?

4. UDP have no mechanisms to integrate reliability, what can you say of protection mechanisms implemented by the applications?

2.2.3 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme4-udp.dmp` (similar to the one previously captured) either from the directory `/Infos/lmd/2022/master/ue/MU4IN001-2022oct`, or from the web page `http://www-npa.lip6.fr/~fourmaux/Traces/labV8.html`, and then analyze it with `wireshark` (without needing administrator rights).

3 Observation of TCP traffic

The aim of this second analysis is to observe the different mechanisms of TCP protocol. For this, we will rely on TCP segments captures.

3.1 Establishment of the TCP connection (without computer assistance)

Here is the first one frame sender when opening a connection:

```
0000   00 50 7f 05 7d 40 00 10   a4 86 2d 0b 08 00 45 00    .P..}@.. ..-...E.

0010   00 3c 17 96 40 00 40 06   6d f3 0a 21 b6 b2 c0 37    .<..@.@. m..!...7

0020   34 28 84 b3 00 50 b6 94   b0 b7 00 00 00 00 a0 02   4(...P.. ........

0030   16 d0 e8 23 00 00 02 04   05 b4 04 02 08 0a 00 6f    ...#.... .......o

0040   a7 21 00 00 00 00 01 03   03 00                      .!...... ..
```

1. Manually analyze (without `wireshark/tshark`) the frame presented above. Directly use the present Lab support to surround the various fields on the capture
2. What are the control bits (TCP flags) located? What do they mean?
3. Identify the hosts involved in this exchange. What are their respective roles in the following?
4. What information can we deduce from the ports numbers contained in the above segments?
5. Remember the operation of TCP sequence numbers. Justify the values ??presented in this segment.
6. Can you see the options in the TCP header? If yes, what do they mean?

3.2 Capture and analysis of a TCP connection

3.2.1 Capturing TCP traffic

The aim of this second traffic capture is to understand TCP. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture the HTTP traffic using `wireshark` or `tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.

- Verify that the HTTP server (`apache2`) is running on `10.0.7.`**N**`3` (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**`2` (window 2)
- Start an HTTP client on `10.0.7.`**N**`1` (window 1)
  - run the client of your choice (`firefox`, `wget`...)
  - establish a connection to the server by navigating to the following URL: `http://10.`**N**`.1.`**N**`3` (display the default page of the Apache server)
- View the `wireshark` window to see the traffic captured
- **Filter the trace to keep only the TCP traffic** (filter = `tcp`). Save the filtered trace for later reuse. Keep the client running so that you can continue to use it during the following analysis.

3.2.2 Analysing the TCP exchange

1. Comparing to the capture of similar HTTP exchanges in **Lab n°2** (with filter = http or tcp.port == 80 and tcp.len > 0), what differences do you notice with the trace **displayed** here?
2. What are the control bits (TCP flags) positioned in the different frames? What do they mean?
3. Check the operation of TCP sequence numbers. **Beware,** `wireshark` **uses a relative numbering**. Compare the actual values read from the frame and those given by `wireshark` for sequence and acknowledgment numbers. Justify the values presented.
4. What can you say about the flow control for the studied segments?
5. Can you find other options in TCP headers? If yes, what do they mean?

3.2.3 Without the testbed...

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme4-tc1.dmp` (similar to the one previously captured) and then analyze it with `wireshark` (without needing administrator rights).

3.3 Analysis of detailed of a TCP exchange

Use the trace backup `tme4-tc1.dmp` proposed above, then scan it onto the PPTI host with `wireshark` started without administrator rights.

Below are displayed the first frames of this trace partially decoded with the `tcpdump` Unix tool (based on `libpcap`, the same library as `wireshark/tshark` capture, but more suited for a textual presentation):

```
00:00:00.000000 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [S], seq 3063197879, win 5840,
[A]                                  options [mss 1460,sackOK,TS val 7317281 ecr 0,nop,wscale 0], length 0

00:00:00.170558 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [S.], seq 610765288, ack 3063197880, win 64240,
[B]                                  options [mss 1402,nop,wscale 0,nop,nop,TS val 0 ecr 0,nop,nop,sackOK], length 0

00:00:00.170618 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 1, win 5840,
[C]                                  options [nop,nop,TS val 7317298 ecr 0], length 0

00:00:00.170819 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [P.], seq 1:486, ack 1, win 5840,
[D]                                  options [nop,nop,TS val 7317298 ecr 0], length 485

00:00:00.370505 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [.], seq 1:1391, ack 486, win 63755,
[E]                                  options [nop,nop,TS val 19332362 ecr 7317298], length 1390

00:00:00.370560 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 1391, win 8340,
[F]                                  options [nop,nop,TS val 7317318 ecr 19332362], length 0

00:00:00.381289 IP 192.55.52.40.80 > 10.33.182.178.33971: Flags [.], seq 1391:2781, ack 486, win 63755,
[G]                                  options [nop,nop,TS val 19332362 ecr 7317298], length 1390

00:00:00.381336 IP 10.33.182.178.33971 > 192.55.52.40.80: Flags [.], ack 2781, win 11120,
[H]                                  options [nop,nop,TS val 7317319 ecr 19332362], length 0

...
```

1. Draw **precisely** the chronogram corresponding to this exchange **(Please observe the time scale for succeed to visualize the evolution the exchanges)**.

2. We would like to study the half-connection corresponding to the emission data from the server (`192.55.52.40.www`) to the client (`10.33.182.178.33971`). Complete the following lines of the table (relative sequence numbers):

action	window's base	sending pointer	window's end	window's size	*comment*
receiving `A`	—	—	—	5840	*win 5840*
sending `B`	0 (610765288)	1	5840	—	*SYN, +1 no sending*
receiving `C`	1	1	5841	5840	*ACK*
receiving `D`	1	1	5841	5840	*ACK*
sending `E`	1	1391	5841	—	
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					
...					

3. Reviewing the evolution of sequence numbers.

4. QWhat can you say about the buffer management?

5. Are you seeing new options? Can you explain them?

6. What can you say aboutof the generation of acknowledgments by the receiver?

7. How ends the communication? Detail the final exchanges.

4 Imbricated TCP exchanges

4.1 Capturing and analyzing of two imbricated TCP connections

4.1.1 Capturing of two imbricated TCP connections

The aim of this last traffic capture is to understand the imbrication of two TCP connections associated to the FTP application. On the networking testbed, again using Topology 1 (client and server on the same LAN), capture HTTP traffic using `wireshark` or `tshark`:

- From PPTI PC **N**, if you do not already have three terminal windows open, connected to the client, monitor, and server VMs, establish these connections now.
- Verify that the FTP server (`ftpd`) is running on `10.0.7.`**N**`3` (window 3)
- Start the capture by running the sniffer on interface `eth1` of host `10.0.7.`**N**`2` (window 2)
- Start an FTP client on `10.0.7.`**N**`1` (window 1)
  - invoke the client at the command line, establishing a connection to the server by typing: `ftp 10.`**N**`.1.`**N**`3` (this is the IPv4 address of the server on the experimental LAN)
  - log in as `etudiant`, with the corresponding password
  - choose a file and download it to the client machine, using the FTP client's `get` command
  - terminate the exchange, using the FTP client's `quit` command
- Observe the trace captured in the `wireshark` window
- **Filter the traffic to keep only TCP** (filter = `tcp`). Save the filtered trace for later reuse. Keep the application running in order to conduct the following analysis.

4.1.2 Analyzing of two imbricated TCP connections

Find the relation between the exchanged messages on the network, on the FTP control connection and those displayed by the application (user interface on the client)..

1. Watch the exchange capture and explain the actions performed at the application level.

2. Draw the chronogram corresponding to this exchanges using a different color per connection.

3. What can you say about the usage of the `PUSH` flag?

4.1.3 Without the testbed. . .

If you have difficulty accessing the networking testbed, or you would simply prefer to work from another machine, you can download the trace `tme4-tc2.dmp` (similar to the one previously captured) and then analyze it with `wireshark` (without needing administrator rights).

5 Before leaving the room

- If you have saved some traces on the monitor VM, do not forget to transfer them back to your PPTI user account. Type the following command on a local terminal of the PPTI host: `scp etudiant@10.0.7.`**N**`2:<trace> <dest>`
- Before closing your connections to the virtual machines, be sure to restore them to the state in which you found them, removing any modifications you might have made.