Transport Layer (Connectionless vs Connection-oriented) – Lab4 Part1

Contents

Introduction

The goal of this lab is to learn how to recognize and analyze connectionless and connection-oriented transport protocols. The lab has several **milestones**. Make sure you reach each one before advancing to the next.

For delivery, submit a PDF report where you answer **only** those steps that are marked with **REPORT(%)**. Additionally, you should submit the code or capture files, if they were **explicitly** asked for. The percent point gives you an indication of the score of that question. Lab4 (both parts) counts for **4.5 points** of your final score in this course.

System Setup

In this lab, you will stream a video from a streaming server (container) and then play the streamed video in your VM, as shown in Figure 1.

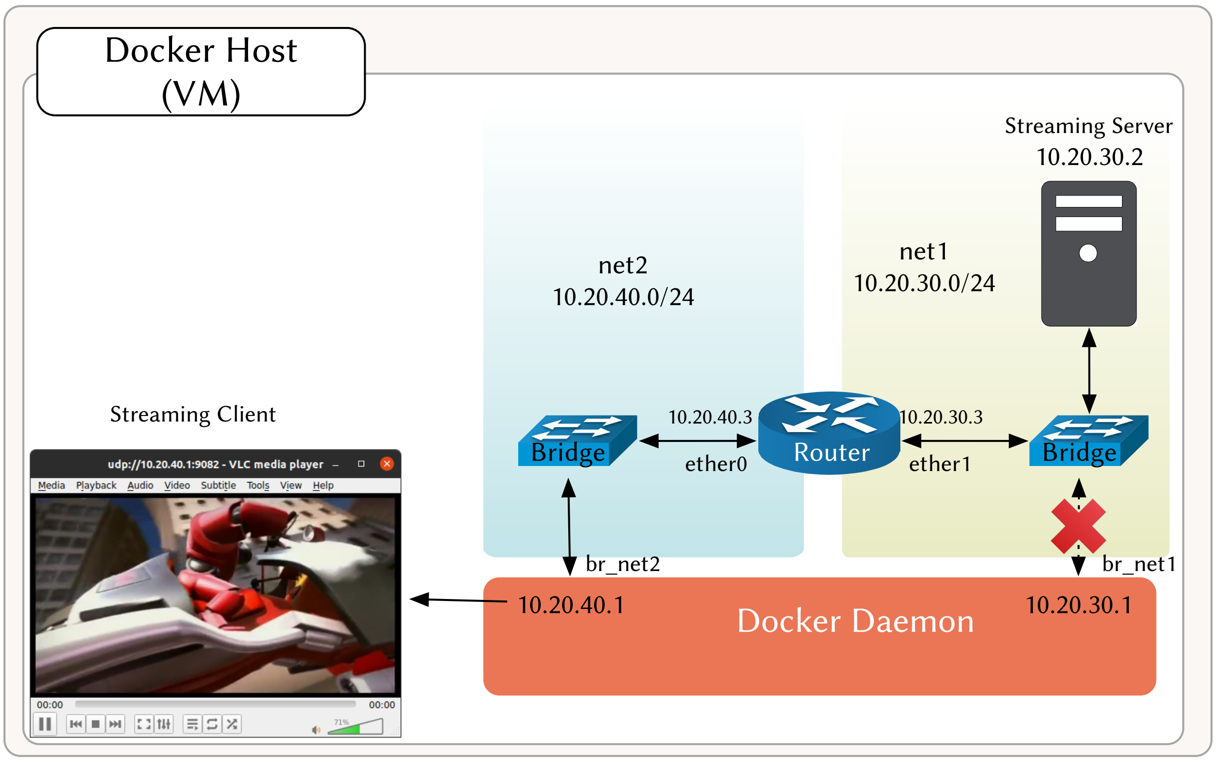

Figure 1: System Setup

- You will use VLC media player to read a video file and stream it over a network. VLC is already installed in your container and in your VM.
- To simulate realistic network scenarios, you will use Linux traffic control tc (https://linux.die.net/

man/8/tc) to introduce packet loss, network delay, packet corruption, and bandwidth limits. For an introduction on how to use the traffic control, refer to the support document.

- You will manipulate traffic control in the "router" container. To ensure that traffic from the streaming server is forwarded through the router, you need to delete the direct connection from "10.20.30.0/29" to the host and add a new route (after starting the containers with docker-compose). This is similar to what you did in TTM4175 (https://ttm4175.iik.ntnu.no/networking-3-2020.html#configuring-simple-routing).

```
docker-compose up -d --build
sudo ip route del 10.20.30.0/29
sudo ip route add 10.20.30.0/29 via 10.20.40.3
```

- Traceroute from your host to the streaming server and make sure that packets are are forwarded through the router.
- Since the traffic control (tc) will also affect `ssh` connection, we recommend to use `docker attach` instead. We also recommend to use `tmux` to access multiple terminal sessions.

  > *HINT: Typing commands in `ssh` will be slow, but correct even when applying packet loss or packet corruption since `ssh` is connection-oriented. If you use `docker attach`, do not kill the container when exiting (by pressing `Ctrl+c` or typing `exit`), use detach instead (press `Ctrl+p` followed by `Ctrl+q`)*

- There are some sample videos to use for streaming in the "server/videos" folder, but feel free use your own video.
- All the commands in this lab assume that your present working directory (`pwd`) is inside "lab4" directory.

1 Milestone 1 – Ideal Network

1.1 Streaming in an Ideal Network

In this milestone you will stream a video over an ideal network scenario (i.e. without packet loss, corruption, delay, or bandwidth limit).

- To stream a video over **UDP (connectionless)**, run the following command in the streaming server (*change the video name*):

```
cvlc ~/work_dir/videos/[video-name.mp4] --loop --ttl 12 --sout \
   '#std{access=udp,mux=ts,dst=10.20.40.1:9082}'
```

- In the docker host, open `vlc` (VLC player) and follow these steps:

1. Navigate to **Media→Open Network Stream**
2. Type “udp://@10.20.40.1:9082” into **network URL**
3. Click in **Play** This will play the streamed video.

- Now you will repeat the same process but using TCP instead. To stream over **TCP (connection-oriented)**, run the following command in the streaming server:

  ```
  cvlc ~/work_dir/videos/[video-name.mp4] --loop --ttl 12 --sout \
   '#std{access=http,mux=ts,dst=10.20.30.2:9080}'
  ```

  This will stream a video using TCP (with HTTP).

- In your host machine, open `vlc` and follow these steps:

  1. Navigate to **Media→Open Network Stream**
  2. Type “http://10.20.30.2:9080” into **network URL**
  3. Click in **Play**

  This will play the streamed video.

1.2 Capturing Packets in an Ideal Network

- Stop the video streaming and start packet capturing in your **VM** on the interface with IP address “10.20.40.1” (“br_net2”):

  ```
  sudo tcpdump -i br_net2 -w server/udp_ideal.pcap
  ```

- Start video streaming over UDP and play the video in ‘vlc’. Stop the streaming after a short amount of time (e.g. after 30 seconds).

  > *HINT: use ‘tmux’ to have two terminals, one for capturing packets and the other for streaming.*

- Repeat the previous two steps but use TCP for streaming. Name your file “tcp_ideal.pcap”.

- You will use these files (“tcp_ideal.pcap” and “udp_ideal.pcap”) in later milestones for comparison.

Q1. **REPORT(3%):** Submit your capture files (“udp_ideal.pcap” and “tcp_ideal.pcap”) along with the report.

2 Milestone 2 – Packet Loss

2.1 Streaming with Packet Loss

- Introduce a packet loss **in the “router”**, on the interface “ether0” (10.20.40.3). You can start with a 5% loss:

```
sudo tc qdisc add dev ether0 root netem loss 5%
```

- Stream a video **over TCP** and see what happens to the video in comparison with the ideal scenario (milestone1). Try to change the amount of packet loss until you see a noticeable difference:

```
Sudo tc qdisc replace dev ether0 root netem loss [amount_of_packet_loss%]
```

- Then Stream a video **over UDP** and see what happens to the video in the presence of packet loss.

> *HINT: You can remove `qdisc` with this command in case you want to restart: `sudo tc qdisc del dev ether0 root`*

Q2. **REPORT(4%):** Briefly describe and compare what happens to the video stream in the presence of packet loss when streamed over UDP vs TCP. In which case the video took longer time to finish?

> *HINT: Try to give reasons based on the protocol fundamentals of UDP vs TCP.*

2.2 Analyzing Packet Loss

Now, you will capture and analyze packets in the of presence packet loss. You will capture packets in the server (before the router, thus before applying packet loss), and in your VM (after the router, thus after applying packet loss).

> *HINT: Refer back to Figure 1.*

2.2.1 TCP

- Start packet capturing at the “server”:

```
sudo tcpdump -i eth0 -w ~/work_dir/tcp_loss_before_router.pcap
```

- In your **VM** start packet capturing on the interface “br_net2”:

```
sudo tcpdump -i br_net2 -w server/tcp_loss_after_router.pcap
```

- Then stream a video over **TCP** in the presence of packet loss. Stop the streaming after a short amount of time (e.g. 30 seconds) and stop the packet capturing.
- Open the capture file (“tcp_ideal.pcap” and “tcp_loss_after_router.pcap”) in Wireshark. Then answer the following questions and validate your answer with an annotated screenshot:

> *HINT: you can apply a display filter to show only the packets exchanged between the server and the client. This filter* `ip.addr eq 10.20.30.2 and ip.addr eq 10.20.40.1` *will display only packets that have both IP addresses ("10.20.30.2" and "10.20.40.1") regardless of the source and destination.*

Q3. **REPORT(4%)** Compare between captured tcp packets in the ideal connection vs lossy connection.

> *HINT:You can apply these display filters:* `tcp.analysis.lost_segment` *to indicate any gap in sequence numbers, and* `tcp.analysis.retransmission` *to display all retransmissions*

- Open the two capture files ("tcp_loss_before_router.pcap" and "tcp_loss_after_router.pcap") in Wireshark. Then answer the following questions and validate your answer with an annotated screenshot:

Q4. **REPORT(4%)** Can you see any difference between captured packets before and after applying packet loss when streaming over TCP? Explain what you see?

> *HINT: In Wireshark navigate to **Analyze→Expert Information**. This will give you a summary of network behavior, thus you can find network problems better than inspecting packet list (https://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html).*

Q5. **REPORT(4%)** Compare the "Expert Information" for the packet capture files of before and after applying packet loss, as well as the ideal connection. Does it make sense considering the protocol fundamentals of TCP?

2.2.2 UDP

- Start packet capturing at the "server":

```
sudo tcpdump -i eth0 -w ~/work_dir/udp_loss_before_router.pcap
```

- In your **VM** start packet capturing on the interface "br_net2":

```
sudo tcpdump -i br_net2 -w server/udp_loss_after_router.pcap
```

- Then stream a video over **UPD** in the presence of packet loss (the same amount of packet loss as you did for TCP). Stop the streaming after a short amount of time (e.g. 30 seconds) and stop the packet capturing.
- Open the two capture files (udp_loss_before_router.pcap and udp_loss_after_router.pcap) in Wireshark. Then answer the following question and validate your answer with an annotated screenshot:

Q6. **REPORT(4%)** Can you see any difference between captured packets before and after applying packet loss when streaming over UPD? Is there any mechanism in the UDP protocol that lets you track packets and detect packet loss (such as sequence number or acknowledgement number)?

- Remove the packet loss in the router:

```
sudo tc qdisc del dev ether0 root
```

3 Milestone 3 – Packet Corruption

3.1 Streaming with Packet Corruption

- Introduce a packet corruption in the “router”, on the interface “ether0”. You can start with a 5%:

```
sudo tc qdisc add dev ether0 root netem corrupt 5%
```

- Stream a video over TCP and see what happens to the video in comparison with the ideal scenario. Try to change the amount of packet corruption until you see a noticeable difference:

```
Sudo tc qdisc replace dev ether0 root netem corrupt [amount_of_packet_corruption%]
```

- Then Stream a video over UDP and see what happens to the video in the presence of packet corruption.

Q7. **REPORT(4%):** Briefly describe and compare what happens to the streamed video in the presence of packet corruption when streamed over UDP vs TCP.

> *HINT: Try to give reasons based on the protocol fundamentals of UDP vs TCP.*

Q8. **REPORT(3%):** Do you think there is a difference between packet corruption and packet loss (both in UDP and TCP)?

3.2 Analyzing Packet Corruption

Q9. **Extra Credit:** Perform the same analysis as you did in (2.2 Analyzing Packet loss), but for packet corruption instead of packet loss. Try to find other analysis tools in Wireshark for packet corruption.

- Remove the packet corruption in the router:

```
sudo tc qdisc del dev ether0 root
```

4 Milestone 4 – Bandwidth Limits

4.1 Streaming with Bandwidth Limits

When analyzing the required bandwidth (maximum transfer capacity of a network), you need to know the throughput of your application. In this milestone, you need to know the amounts of packets produced by the streaming server (in bits/second). You can use Wireshark I/O Graph (https://www.wireshark.org/docs/wsug_html_chunked/ChStatIOGraphs.html) to see the throughput (as a graph over time).

Q10. **REPORT(4%)** Determine (approximately) the average throughput of your video streaming over UDP and TCP in the **ideal** network. Provide a screenshot of the throughput graph.

> *HINT: First you need to apply the display filter* `ip.addr eq 10.20.30.2 and ip.addr eq 10.20.40.1`*, then navigate to **Statistics→I/O Graph**. In the lower part of the graph, select only "Filtered packets". In the "Y Axis" tap,select "Bits" to show Bits/sec instead of Packets/sec.*

- Introduce a bandwidth limit in the "router", on the interface "ether0". You can start with a value less than the average throughput from above (e.g. in my case 500kbit):

```
sudo tc qdisc add dev ether0 root tbf rate 500kbit burst 500kbit latency 50ms
```

- Stream a video over TCP and see what happens to the video in comparison with the ideal scenario. Try to change the amount of bandwidth until you see a noticeable difference:

```
sudo tc qdisc replace dev ether0 root tbf rate [banwidth_limit]kbit \
   burst [bandwidth_limit]kbit latency 50ms
```

- Then stream a video over UDP and see what happens to the video in the presence of bandwidth limits.

Q11. **REPORT(4%):** Briefly describe and compare what happens to the streamed video in the presence of bandwidth limits when streamed over UDP vs TCP.

> *HINT: Try to give reasons based on the protocol fundamentals of UDP vs TCP.*

4.2 Analyzing Bandwidth Limits

4.2.1 TCP

- Start packet capturing at the "server":

```
sudo tcpdump -i eth0 -w ~/work_dir/tcp_bandwidth_before_router.pcap
```

- In your **VM** start packet capturing on the interface "br_net2":

```
sudo tcpdump -i br_net2 -w server/tcp_bandwidth_after_router.pcap
```

- Then stream a video over **TCP** in the presence of bandwidth limits. Stop the streaming after a short amount of time (e.g. 30 seconds) and stop the packet capturing.

- Open the capture file ("tcp_ideal.pcap" and "tcp_bandwidth_after_router.pcap") in Wireshark. Then answer the following questions and validate your answer with an annotated screenshot:

Q12. **REPORT(4%)** Compare the throughput (using I/O Graph) between the TCP stream in the ideal connection vs bandwidth-limited connection.

Q13. **REPORT(4%)** Provide a comparison between the server window size in the ideal connection vs bandwdith-limited connection. Briefly explain how TCP adapts to bandwidth limit using "receive window".

> *HINT: You can use Wireshark window scaling graph to show a plot of window size over time. First apply the display filter* `ip.src == 10.20.30.2 and ip.dst ==10.20.40.1`*, then navigate to* ***Statistics→TCP Streams Graphs→Window Scaling****.*

- Open the two capture files ("tcp_bandwidth_before_router.pcap" and "tcp_bandwidth_after_router.pcap") in Wireshark. Then answer the following questions and validate your answer with an annotated screenshot:

Q14. **REPORT(4%)** Can you see any difference between throughput (using I/O Graph) before and after applying bandwidth limit when streaming over TCP? Explain what you see?

Q15. **REPORT(4%)** Compare the "Expert Information" between packet capture files of before and after applying bandwidth limit, as well as the ideal connection.

4.2.2 UDP

- Start packet capturing at the "server":

```
sudo tcpdump -i eth0 -w ~/work_dir/udp_bandwidth_before_router.pcap
```

- In your **VM** start packet capturing on the interface "br_net2":

```
sudo tcpdump -i br_net2 -w server/udp_bandwidth_after_router.pcap
```

- Then stream a video **over UDP** in the presence of bandwidth limit. Stop the streaming after a short amount of time (e.g. 30 seconds) and stop the packet capturing.
- Open the two capture files ("udp_bandwidth_before_router.pcap" and "udp_bandwidth_after_router.pcap") in Wireshark. Then answer the following questions and validate your answer with an annotated screenshot:

Q16. **REPORT(4%)** Compare between the throughput (using I/O Graph) of the UDP stream before and after applying bandwidth limit.

4.2.3 UDP vs TCP

Q17. **REPORT(4%)** Compare between the throughput (using I/O Graph) of the UDP stream vs TCP stream in both cases before and after applying bandwidth limit.

- Remove the bandwidth limit in the router:

```
sudo tc qdisc del dev ether0 root
```

5 Milestone 4 – Network Delay

5.1 Streaming with Network Delay

- Introduce a network delay in the router.

```
sudo tc qdisc add dev ether0 root netem delay 200ms
```

- Stream a video over TCP and see what happens. Try different values of network delay:

```
sudo tc qdisc replace dev ether0 root netem delay [delay_value]ms
```

- Then stream a video over UDP and see what happens.

Q18. **REPORT(3%):** Briefly describe and compare what happens to the streamed video in the presence of network delay when streamed over UDP vs TCP. Is this an expected behavior?

5.2 Analyzing Network Delay

Q19. **Extra Credit:** Perform network delay analysis similar to what you did in Milestone 2 and 4. You would have to use different analysis tools in Wireshark more appropriate for analyzing network delay.

Contributors

Abdulmajid Murad, David Palma, Stanislav Lange, Mathias Pettersen, Sebastian Fuglesang.