

CPSC 644 - Computer Networks

Assignment 1

Submitted by

Student ID	Name
230171256	Simon Kraft

February 11, 2026

Exercise 1

Consider a packet of length L that begins at end system A and travels over three links to the destination end system. These three links are connected by *two* packet switches. Let d_i , s_i , and R_i denote the length, propagation speed, and the transmission rate of link $i \in \{1, 2, 3\}$. The packet switches delay each packet by d_{proc} .

Hint: the processing delay is the same for each switch and there are no queueing delays.

a)

Question: What is the total end-to-end delay in terms of d_i , s_i , R_i and d_{proc} ?

The network considered in this exercise is drawn in Figure 1.

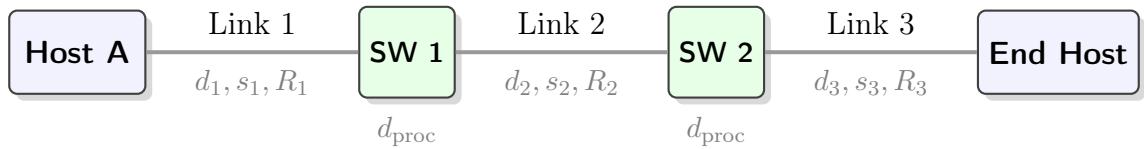


Figure 1: Drawing of Network Topology

To calculate the end-to-end delay, we need to calculate the delay for each single hop on the way from Host A to the End Host. This nodal delay can be expressed by the formula:

$$d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

where:

- d_{proc} : nodal processing delay (e.g. check for bit errors, check headers)
- d_{queue} : queueing delay (waiting at output link for transmission)
- d_{trans} : transmission delay (time to transmit packet onto link)
- d_{prop} : propagation delay (time for packet to completely propagate from one hop to next)

Furthermore, transmission and propagation delay can be calculated using the above given link lengths d_i , propagation speeds s_i , and transmission rates R_i for all links $i \in \{1, 2, 3\}$ and the packet length L .

$$d_{\text{trans}}^{(i)} = \frac{L}{R_i} \quad d_{\text{prop}}^{(i)} = \frac{d_i}{s_i}$$

As processing delay d_{proc} is the same for each packet switch and there are no queueing delays, the formula to calculate the end-to-end delay in this scenario is given by:

$$\begin{aligned}
d_{\text{end-to-end}} &= 2 \cdot d_{\text{proc}} + \sum_{i=1}^3 \frac{L}{R_i} + \sum_{i=1}^3 \frac{d_i}{s_i} \\
d_{\text{end-to-end}} &= 2 \cdot d_{\text{proc}} + \sum_{i=1}^3 d_{\text{trans}}^{(i)} + \sum_{i=1}^3 d_{\text{prop}}^{(i)}
\end{aligned} \tag{1}$$

The transmission delay needs to be included for each link, as the links do not directly push incoming bits through. Instead, the routers work in a store-and-forward way, meaning the entire packet needs to arrive at router before it can be transmitted on the next link.

b)

Question: For the given values, what is the end-to-end delay?

$$\begin{aligned}
L &= 1,500 \text{ bytes} = 12,000 \text{ bits} \\
\forall i \in \{1, 2, 3\} : s_i &= 2.5 \cdot 10^8 \text{ m/s} = 2.5 \cdot 10^5 \text{ km/s} \\
\forall i \in \{1, 2, 3\} : R_i &= 2.5 \text{ Mbps} = 2,500,000 \text{ bps} \\
d_{\text{proc}} &= 3 \text{ ms} = 0.003 \text{ s} \\
d_1 &= 5,000 \text{ km} \quad d_2 = 4,000 \text{ km} \quad d_3 = 1,000 \text{ km}
\end{aligned}$$

To calculate the end-to-end delay we use Equation 1 from the previous exercise:

$$\begin{aligned}
d_{\text{end-to-end}} &= 2 \cdot 0.003 \text{ s} + 3 \cdot \frac{12,000 \text{ bits}}{2,500,000 \text{ bits/s}} + \frac{5,000 \text{ km} + 4,000 \text{ km} + 1,000 \text{ km}}{2.5 \cdot 10^5 \text{ km/s}} \\
&= 0.006 \text{ s} + 0.0144 \text{ s} + 0.04 \text{ s} = \mathbf{0.0604 \text{ s}}
\end{aligned}$$

The end-to-end delay in this scenario is 0.0604 seconds.

c)

Question: What is the end-to-end delay formula, when there is no processing delay, the transmission rate is the same for all hops (R), and the packet switch doesn't store-and-forward?

As a receiver of data can immediately start transmitting the bits it received again onto the next link, there is only the transmission delay from the Host A to push all the bits of the packet onto the first link. Therefore, the other transmission delays are not existent in this scenario.

Furthermore, as there is no processing delay, i.e. $d_{\text{proc}} = 0$, the final formula changes to:

$$d_{\text{end-to-end}} = 2 \cdot d_{\text{proc}} + \frac{L}{R} + \sum_{i=1}^3 \frac{d_i}{s_i} \quad (2)$$

$$d_{\text{end-to-end}} = \frac{L}{R} + \sum_{i=1}^3 \frac{d_i}{s_i}$$

Exercise 2

In modern packet-switched networks, including the Internet, the source host segments long, application-layer messages (for example, an image or a music file) into smaller packets and sends the packets into the network. The receiver then reassembles the packets back into the original message. We refer to this process as message segmentation.

Figure 2 shows the end-to-end transport with and without message segmentation. In this example, the message L sent from source to destination is 10^6 bits long and each link has 5 Mbps. Propagation, queueing, and processing delay are ignored.

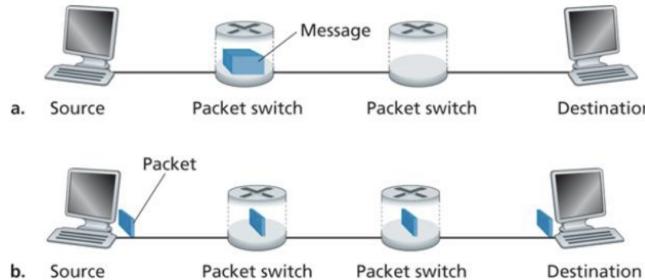


Figure 2: End-to-end message transport. a) without message segmentation and b) with message segmentation.

a)

- Question:** How long does it take to move the message from the source host to the first packet switch without message segmentation?

Information that we have:

- Transmission happens at $R = 5 \text{ Mbps} = 5,000,000 \text{ bits/s}$.
- Store-and-forward is used, i.e. entire packet must arrive at switch before forwarding
- Formula for calculating transmission delay $\frac{L}{R}$

- $L = 10^6$ bits = 1,000,000 bits

Therefore, moving from source to the first packet switch takes:

$$d_{1\text{-hop}} = \frac{L}{R} = \frac{1,000,000 \text{ bits}}{5,000,000 \text{ bits/s}} = \mathbf{0.2 \text{ s}}$$

2. Question: What is the total time to move the message from source host to destination host?

As we have have 3 links, the whole packet needs to be transmitted 3 times. Therefore the total is given by:

$$d_{\text{total}} = 3 \cdot \frac{L}{R} = 3 \cdot \frac{1,000,000 \text{ bits}}{5,000,000 \text{ bits/s}} = \mathbf{0.6 \text{ s}}$$

b)

Setup: Message segmented in 100 packets, each size 10,000 bits.

1. Question: How long does it take to move the first packet from source host to the first switch?

As the transmission rate remains equal and the number of bits reduced to 10,000, the result is:

$$d_{\text{first-packet-at-1-hop}} = \frac{L}{R} = \frac{10,000 \text{ bits}}{5,000,000 \text{ bits/s}} = \mathbf{0.002 \text{ s}}$$

2. Question: At what time will the second packet be fully received at the first switch?

As the transmission of the second packet begins, when the first packet is fully transmitted (after 0.002 seconds), we just need to add the transmission time for the second packet (also 10,000 bits large) onto that:

$$d_{\text{second-packet-at-1-hop}} = 0.002 + \frac{L}{R} = 0.002 + \frac{10,000 \text{ bits}}{5,000,000 \text{ bits/s}} = \mathbf{0.004 \text{ s}}$$

c)

Question: How long does it take to move the file from source host to the destination host when message segmentation is used?

To answer this, we need to determine, when the last packet gets transmitted by the source host and sum it up with the time, it takes for the last packet to go through all 3 links to the destination host:

$$\begin{aligned}
d_{\text{last-packet-at-destination}} &= \underbrace{99 \cdot \frac{10,000 \text{ bits}}{5,000,000 \text{ bits/s}}}_{\text{Point of time, when transmission of last packet starts}} + \underbrace{3 \cdot \frac{10,000 \text{ bits}}{5,000,000 \text{ bits/s}}}_{\text{3 transmission delays until last packet arrives at destination}} \\
&= 102 \cdot 0.002 \text{ s} = \mathbf{0.204 \text{ s}}
\end{aligned}$$

Comparison: of this result with the result from section a).

With message segmentation we are able to reduce the time it takes to transmit the whole data to the destination from 0.6 seconds to 0.204 seconds. This equals to a reduction of end-to-end delay by almost 66%.

In general, increasing the number of hops also increases the benefits of message segmentation. Without message segmentation, each hop adds another $\frac{L}{R}$ seconds to the total end-to-end delay. With message segmentation, each hop adds only $\frac{l}{R}$ seconds where l is only the size of one single packet. Conversely, decreasing the number of hops, reduces this advantage of message segmentation. With only a single link, sending a message with and without segmentation takes equally long.

d)

Advantages of Message Segmentation

- **Error Recovery:** when the buffer of a packet switch is full or when there is a bit error in an incoming packet, this packet gets dropped. This can especially happen if large packets arrive at the switch. Dropped packets will then need to be retransmitted. With message segmentation, the system only needs to worry about the specific packet that got dropped, not about the entire message.
- **Sharing Switches:** A single large message blocks the link for a long time. During this time the traffic from other hosts cannot be served. With message segmentation, packets from different sources can be interleaved more easily, so that more people can fairly share the resources of the switch.

e)

Disadvantages of Message Segmentation

- **Encapsulation:** Each packet must have its own header attached to it when it goes down the protocol stack. Sending all these headers increases the total amount of data to be sent, bandwidth is therefore wasted on administrative data rather than on actual payload data.
- **Message Reassembly:** Not all packets of a segmented message take the same path throughout the network. Therefore, the destination host must use compute power and memory to hold this data in a buffer and to later reassemble it correctly before an application can use them.

Exercise 3

Perform a traceroute between source and destination on the same continent at three different hours of the day.

5 PM

```
-> traceroute -I harvard.edu
traceroute to harvard.edu (192.0.66.20), 64 hops max, 48 byte packets
 1 142.207.103.1 (142.207.103.1) 88.418 ms 22.988 ms 25.163 ms
 2 10.101.1.49 (10.101.1.49) 12.388 ms 5.706 ms 14.614 ms
 3 10.101.2.41 (10.101.2.41) 3.689 ms 3.924 ms 3.166 ms
 4 * 204.239.83.195 (204.239.83.195) 3.794 ms 3.176 ms
 5 775-tx-unbc.pgrgl.bc.net (207.23.242.21) 3.387 ms 3.442 ms 3.663 ms
 6 * * *
 7 rdihrt-be31-100.ok.shawcable.net (66.163.72.189) 13.492 ms 11.647 ms 11.776 ms
 8 rdics-be5.ok.shawcable.net (66.163.72.253) 14.143 ms 13.643 ms 13.547 ms
 9 * * rc2st-be1.vc.shawcable.net (66.163.72.233) 186.513 ms
10 rc2wt-be50-1.wa.shawcable.net (66.163.70.186) 23.494 ms 23.059 ms 23.380 ms
11 rc1wt-be18-1.wa.shawcable.net (66.163.64.81) 24.186 ms 23.563 ms 65.697 ms
12 six.automatic.net (206.81.81.78) 23.679 ms 23.432 ms 121.339 ms
13 192.0.66.20 (192.0.66.20) 23.190 ms 39.053 ms 23.580 ms
-> | 31s base 05:34:01 PM
```

11 PM

```
-> traceroute -I harvard.edu
traceroute to harvard.edu (192.0.66.20), 64 hops max, 48 byte packets
 1 142.207.57.65 (142.207.57.65) 19.392 ms 21.167 ms 21.691 ms
 2 10.101.1.97 (10.101.1.97) 4.575 ms 8.696 ms 8.306 ms
 3 10.101.2.41 (10.101.2.41) 3.418 ms 3.011 ms 2.879 ms
 4 204.239.83.195 (204.239.83.195) 3.559 ms 3.488 ms 6.046 ms
 5 775-tx-unbc.pgrgl.bc.net (207.23.242.21) 3.912 ms 3.532 ms 3.471 ms
 6 * * *
 7 rdihrt-be31-100.ok.shawcable.net (66.163.72.189) 13.724 ms 13.051 ms 12.165 ms
 8 rdics-be5.ok.shawcable.net (66.163.72.253) 13.871 ms 13.799 ms 13.897 ms
 9 rc2st-be1.vc.shawcable.net (66.163.72.233) 20.723 ms 18.281 ms 22.901 ms
10 rc2wt-be50-1.wa.shawcable.net (66.163.70.186) 23.669 ms 23.426 ms 25.343 ms
11 rc1wt-be18-1.wa.shawcable.net (66.163.64.81) 24.025 ms 23.553 ms 26.353 ms
12 six.automatic.net (206.81.81.78) 23.576 ms 23.338 ms 23.807 ms
13 192.0.66.20 (192.0.66.20) 23.680 ms 23.265 ms 23.120 ms
-> | 16s base 11:38:12 PM
```

12 PM

```
-> traceroute -I harvard.edu
traceroute to harvard.edu (192.0.66.20), 64 hops max, 48 byte packets
 1 142.207.103.1 (142.207.103.1) 29.284 ms 22.166 ms 24.984 ms
 2 10.101.1.49 (10.101.1.49) 9.180 ms 10.122 ms 15.344 ms
 3 10.101.2.41 (10.101.2.41) 6.524 ms 5.941 ms 3.440 ms
 4 * * *
 5 * 775-tx-unbc.pgrgl.bc.net (207.23.242.21) 188.097 ms 1109.013 ms
 6 * * *
 7 rdihrt-be31-100.ok.shawcable.net (66.163.72.189) 625.637 ms 218.311 ms 312.799 ms
 8 rdics-be5.ok.shawcable.net (66.163.72.253) 278.658 ms 147.234 ms 51.146 ms
 9 rc2st-be1.vc.shawcable.net (66.163.72.233) 33.600 ms 24.685 ms *
10 rc2wt-be50-1.wa.shawcable.net (66.163.70.186) 567.120 ms 807.423 ms 45.791 ms
11 * rc1wt-be18-1.wa.shawcable.net (66.163.64.81) 188.922 ms 28.773 ms
12 six.automatic.net (206.81.81.78) 182.356 ms 23.546 ms 128.743 ms
13 192.0.66.20 (192.0.66.20) 41.508 ms 38.641 ms 26.387 ms
-> | 51s base 12:23:37 PM
```

Figure 3: Traceroute of harvard.edu for three different hours of the day.

a)

The sample average and standard deviation are given by:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^3 x_i \quad s = \sqrt{\frac{1}{n-1} \sum_{i=1}^3 (x_i - \bar{x})^2}$$

With this, we can calculate the following:

$$\bar{x}_{5\text{-PM}} = \frac{23.190 + 39.053 + 23.580}{3} = 28.608$$

$$s_{5\text{-PM}} = \sqrt{\frac{(23.190 - 28.608)^2 + (39.053 - 28.608)^2 + (23.580 - 28.608)^2}{2}} = 9.048$$

$$\bar{x}_{11\text{-PM}} = \frac{23.680 + 23.265 + 23.120}{3} = \mathbf{23.355}$$

$$s_{11\text{-PM}} = \sqrt{\frac{(23.680 - 23.355)^2 + (23.265 - 23.355)^2 + (23.120 - 23.355)^2}{2}} = \mathbf{0.291}$$

$$\bar{x}_{12\text{-PM}} = \frac{41.508 + 38.641 + 26.387}{3} = \mathbf{35.512}$$

$$s_{12\text{-PM}} = \sqrt{\frac{(41.508 - 35.512)^2 + (38.641 - 35.512)^2 + (26.387 - 35.512)^2}{2}} = \mathbf{8.031}$$

b)

Number of routers in the path for each of the three hours:

- **5 PM:** 13 routers
- **11 PM:** 13 routers
- **12 PM:** 13 routers

For each of the three hours, the traceroute command passes through 13 routers until it reaches its destination.

c)

Number of ISPs

As the routing for the three different hours is always quite similar, we can assume the **4** different ISPs:

- **Hop 1-4:** Local institutional network of UNBC
- **Hop 5:** BCNET (network for higher education and research in BC)
- **Hop 6:** * * * the router did not respond to traceroute (often due to security reasons)(no ISP)
- **Hop 7-11:** Shawcable commercial ISP operator
- **Hop 12-13:** Automattic network of destination

Largest Delay

The largest increases in delay actually seems to appear when going from the BCNET ISP to the Shawcable ISP. Within the Shawcable ISP there are only small increases in the delay, that can be explained through the large distance between British Columbia and Harvard University. Furthermore, the delays in the 12 PM screenshot vary a lot, e.g. one delay at the BCNET ISP takes around 1100 ms, where as the final delay at the destination is still between 26 and 42 ms.

d)

Perform a traceroute between source and destination on a different continent at three different hours of the day.

5 PM

```
~ > traceroute -I uni-bonn.de
traceroute to uni-bonn.de (131.220.250.29), 64 hops max, 48 byte packets
 1 142.207.1.103.1 (142.207.103.1) 21.999 ms 23.143 ms 8.657 ms
 2 10.101.1.49 (10.101.1.49) 8.393 ms 7.032 ms 6.747 ms
 3 10.101.2.41 (10.101.2.41) 3.458 ms 3.262 ms 3.251 ms
 4 204.239.83.195 (204.239.83.195) 3.543 ms 5.522 ms 3.140 ms
 5 781-oran-unbc.pgrgl.bc.net (207.23.242.37) 5.048 ms 3.465 ms 3.314 ms
 6 205.189.32.45 (205.189.32.45) 9.992 ms 11.295 ms 11.778 ms
 7 toro1rrt1.network.canarie.ca (205.189.32.44) 16.899 ms 15.800 ms 119.823 ms
 8 rgnalrrt3.network.canarie.ca (205.189.32.234) 25.008 ms 24.356 ms 24.319 ms
 9 wmpg2rrt3.network.canarie.ca (205.189.32.239) 31.010 ms 30.862 ms 32.940 ms
10 toro3rrt3.network.canarie.ca (205.189.32.210) 53.184 ms 52.469 ms 52.459 ms
11 toro1rrt3.network.canarie.ca (205.189.32.119) 53.263 ms 52.987 ms 133.331 ms
12 nycn1rrt3.network.canarie.ca (208.75.75.75) 63.647 ms 63.109 ms 63.372 ms
13 canarie.rt0.par.fr.geant.net (62.40.124.221) 130.702 ms 190.059 ms 130.990 ms
14 * * *
15 * * *
16 dfn-gw.rte1.fra.de.geant.net (62.40.124.218) 330.522 ms 146.117 ms 145.608 ms
17 kr-boni68-0.x-win.dfn.de (188.1.236.42) 148.966 ms 148.321 ms 148.438 ms
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *
31 * * *
32 * * *
33 *^C
```

base 05:35:21 PM

x INT 4m 26s base 05:39:50 PM

11 PM

```
~ > traceroute -I uni-bonn.de
traceroute to uni-bonn.de (131.220.250.29), 64 hops max, 48 byte packets
 1 142.207.57.65 (142.207.57.65) 56.590 ms 19.441 ms 21.274 ms
 2 10.101.1.97 (10.101.1.97) 4.520 ms 5.706 ms 8.654 ms
 3 10.101.2.41 (10.101.2.41) 12.281 ms 3.096 ms 3.004 ms
 4 204.239.83.195 (204.239.83.195) 3.634 ms 3.262 ms 3.363 ms
 5 781-oran-unbc.pgrgl.bc.net (207.23.242.37) 3.796 ms 3.331 ms 3.533 ms
 6 205.189.32.45 (205.189.32.45) 9.572 ms 9.415 ms 9.288 ms
 7 toro1rrt1.network.canarie.ca (205.189.32.44) 16.476 ms 16.167 ms 15.982 ms
 8 rgnalrrt3.network.canarie.ca (205.189.32.234) 24.798 ms 24.522 ms 24.498 ms
 9 wmpg2rrt3.network.canarie.ca (205.189.32.239) 30.831 ms 30.452 ms 30.436 ms
10 toro3rrt3.network.canarie.ca (205.189.32.210) 52.945 ms 100.866 ms 53.565 ms
11 toro1rrt3.network.canarie.ca (205.189.32.119) 53.690 ms 53.210 ms 52.990 ms
12 nycn1rrt3.network.canarie.ca (208.75.75.75) 63.654 ms 63.087 ms 63.013 ms
13 canarie.rt0.par.fr.geant.net (62.40.124.221) 130.101 ms 129.735 ms 129.416 ms
14 * * *
15 * * *
16 cr-fra1.x-win.dfn.de (62.40.124.218) 150.791 ms 149.370 ms 149.254 ms
17 kr-boni68-0.x-win.dfn.de (188.1.236.42) 183.623 ms 154.111 ms 151.928 ms
18 * * *
19 *^C
```

base 11:49:48 PM

x INT 53s base 11:50:44 PM

1 PM

```
~ > traceroute -I uni-bonn.de
traceroute to uni-bonn.de (131.220.250.29), 64 hops max, 48 byte packets
 1 142.207.124.1 (142.207.124.1) 99.550 ms 20.000 ms 37.823 ms
 2 10.101.1.81 (10.101.1.81) 5.646 ms 6.687 ms 8.284 ms
 3 10.101.2.41 (10.101.2.41) 3.348 ms 2.917 ms 3.526 ms
 4 204.239.83.195 (204.239.83.195) 3.745 ms 3.400 ms 3.121 ms
 5 781-oran-unbc.pgrgl.bc.net (207.23.242.37) 3.766 ms 3.429 ms 3.269 ms
 6 205.189.32.45 (205.189.32.45) 11.526 ms 9.452 ms 9.817 ms
 7 toro1rrt1.network.canarie.ca (205.189.32.44) 16.588 ms 16.564 ms 16.135 ms
 8 rgnalrrt3.network.canarie.ca (205.189.32.234) 25.292 ms 25.388 ms 99.994 ms
 9 wmpg2rrt3.network.canarie.ca (205.189.32.239) 31.764 ms 31.046 ms 30.514 ms
10 toro3rrt3.network.canarie.ca (205.189.32.210) 53.185 ms 52.814 ms 52.440 ms
11 toro1rrt3.network.canarie.ca (205.189.32.119) 53.763 ms 53.545 ms 89.768 ms
12 nycn1rrt3.network.canarie.ca (208.75.75.75) 63.515 ms 64.437 ms 63.034 ms
13 canarie.rt0.par.fr.geant.net (62.40.124.221) 130.268 ms 130.044 ms 129.447 ms
14 * * *
15 * * *
16 dfn-gw.rte1.fra.de.geant.net (62.40.124.218) 150.958 ms 149.227 ms 175.581 ms
17 kr-boni68-0.x-win.dfn.de (188.1.236.42) 149.186 ms 148.691 ms 148.644 ms
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
^C
```

base 01:02:18 PM

x INT 1m 54s base 01:04:15 PM

Figure 4: Traceroute of uni-bonn.de for three different hours of the day.

Averages and standard deviations

$$\bar{x}_{5-PM} = \frac{148.966 + 148.321 + 148.438}{3} = \mathbf{148.575}$$

$$s_{5-PM} = \sqrt{\frac{(148.966 - 148.575)^2 + (148.321 - 148.575)^2 + (148.438 - 148.575)^2}{2}} = \mathbf{0.344}$$

$$\bar{x}_{11-PM} = \frac{183.623 + 154.111 + 151.92}{3} = \mathbf{163.218}$$

$$s_{11-PM} = \sqrt{\frac{(183.623 - 163.218)^2 + (154.111 - 163.218)^2 + (151.92 - 163.218)^2}{2}} = \mathbf{17.705}$$

$$\bar{x}_{1-PM} = \frac{149.186 + 148.691 + 148.644}{3} = \mathbf{148.8403}$$

$$s_{1-PM} = \sqrt{\frac{(149.186 - 148.8403)^2 + (148.691 - 148.8403)^2 + (148.644 - 148.8403)^2}{2}} = \mathbf{0.300}$$

Number of routers

- **5 PM:** terminated after 33 routers but only until the 17th there were useful responses (further packets probably lost in firewall)
- **11 PM:** terminated after 19 routers but last useful response at 17th router
- **1 PM:** terminated after 22 routers but last useful at 17th router

For each of the three hours, we seemed to have reached our destination (dfn.de) after 17 routers. After that we are just stuck in a firewall and the traceroute packets are not returning to us anymore.

Number of ISPs

As the routing for the three different hours is quite similar, we can assume the **5** different ISPs:

- **Hop 1-4:** Local institutional network of UNBC
- **Hop 5:** BCNET
- **Hop 6-12:** Canarie ISP (formerly Canadian Network for the Advancement of Research, Industry and Education)
- **Hop 13-15/16:** GEANT ISP (European Academic Network)
- **Hop 16/17-end:** DFN (German Research Network)

Largest Delay

The largest increase in delay is observed when going from the Canarie ISP in New York City to the GEANT ISP, by almost doubling the delay due to the trans-oceanic link between North America and Europe which causes a "long" propagation time. The delays within Europe don't increase too much.

Exercise 4

Use the HTTP/1.1 specification in RFC 2616 [1] to answer the following questions.

a)

Mechanism between Client and Server to close a persistent connection

- “If either the client or the server sends the close token in the connection header, that request becomes the last one for that connection” found on page 44 [1]
- Once the closing header is sent, both parties understand that the TCP connection should be closed after the current message got fully transmitted

Who can choose to close the connection?

- Both parties can always send this header to signal the termination of the connection from their side, it is not restricted to only the server or only the client
- Client might terminate the connection when it has finished all planned requests for this particular server
- Server might terminate the connection when the connection has been idle for too long, so that it just takes up valuable resources to keep it open the whole time or same way if the server is approaching its maximum number of open connections

b)

What encryption services are provided by HTTP?

- HTTP does not provide any encryption, it was designed as a plain text protocol where information is broken into packets of data that can easily be sniffed using free software
- All communications in HTTP occur in plain text, including passwords, making them highly accessible and vulnerable for man-in-the-middle attacks
- Every ISP on the way from the destination server to the source host can sniff and read the traffic, it can even inject content into the webpages without approval of the website owner (commonly in the form of advertising) [2]
- HTTPS adds encryption so that the ISP can no longer sniff into the packet and maybe insert ads into it

c)

Can a client open three or more simultaneous connections with a given server according to the RFC standard?

- No, according to this RFC, in HTTP/1.1 a Client should limit its number of simultaneous connections to 2 to a given server
- “Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy” found on page 46 [1]

d)

Is it possible that one side starts closing a connection while the other side is transmitting data via this connection?

- Yes, it is possible that one side decides to close the connection while at the same time the other side tries to do a request, this occurrence is called ”asynchronous close event”
- Both Clients and Servers should always watch whether the other side tries to end the connection at some point
- No matter if client, server or proxy, everyone should be able to recover from such an asynchronous close event
 - Client software should just try to automatically reopen the connection and send the sequence of aborted requests again, if this sequence is idempotent, meaning that the side effects of performing $N > 0$ requests are the same as for performing only a single request, the methods GET, HEAD, PUT, and DELETE fulfill this property.
 - Non-idempotent methods must not automatically do this, but can ask the human operator whether he wants to submit the request again
- Because of the propagation delay between Client and Server, this event can happen, where both parties at the same time decide to make a move
- ”A client, server, or proxy MAY close the transport connection at any time. For example, a client might have started to send a new request at the same time that the server has decided to close the "idle" connection. From the server’s point of view, the connection is being closed while it was idle, but from the client’s point of view, a request is in progress.” found on page 46 [1]

Exercise 5

In this problem, we use the useful `dig` tool to explore the hierarchy of DNS servers. A DNS server in the DNS hierarchy delegates a DNS query to a DNS server lower in the hierarchy, by sending back to the DNS client the name of that lower-level DNS server.

a)

Initiate a sequence of queries, starting from one of the root servers ([a-m].root-servers.net, for the IP address of your UNBC's webserver (www.unbc.ca).

Figure 5: Dig trace for UNBC webserver

Figure 5 shows the delegation chain:

- **Root DNS Server:** a.root-servers.net with IP address 198.41.0.4
 - **TLD DNS Server:** any.ca-servers.ca with IP address 199.4.144.2
 - **Authoritative DNS Server:** ns1.d-zone.ca with IP address 162.219.54.2

Alternative Approach

Iteratively, query each server manually (use `@server` option of dig to include either the hostname or IP address), by the IP address provided from the query to the server higher in the hierarchy. Start with a query to the root server.

Root DNS Server

```

[+] dig @0.0.0.1 servers.net +www.cs

;--> SOA servers.net <--> 0.0.0.1-servers.net www.cs
[+] 1 server found
[+] search options: +noall +nocomm
[+] Got answer from 0.0.0.1-servers.net. [status: NOERROR, id: 37743]
[+] flags: qr rd ra
[+] query: www.cs IN A
[+] WARNING: recursion requested but not available

; OPT PREPROCESSOR:
[+] OPT PREPROCESSOR: 0.0.0.1-servers.net flags: urps
[+] QUESTION SECTION:
    www.cs. IN A

;+> AUTHORITY SECTION:
[+] 0.0.0.1-servers.cs. 177886 IN NS d.cs-servers.cs.
[+] 0.0.0.1-servers.cs. 177886 IN NS b.cs-servers.cs.
[+] 0.0.0.1-servers.cs. 177886 IN NS c.cs-servers.cs.
[+] 0.0.0.1-servers.cs. 177886 IN NS j.cs-servers.cs.

;+> ADDITIONAL SECTION:
[+] 0.cs-servers.cs. 177886 IN A 45.142.229.181
[+] 0.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100
[+] www.cs-servers.cs. 177886 IN A 197.14.1.62
[+] www.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100
[+] 0.cs-servers.cs. 177886 IN A 197.14.1.63
[+] 0.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100/2
[+] 1.cs-servers.cs. 177886 IN A 197.14.1.64
[+] 1.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100/2
[+] 2.cs-servers.cs. 177886 IN A 197.14.1.65
[+] 2.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100/2
[+] 3.cs-servers.cs. 177886 IN AAAA 2001:4810::1:100/2

[+] Query time: 88 msec
[+] EDNS(0): 0, payload: 4096, flags: RD, Z, A
[+] WHEN: Wed Feb 13 13:35:15 PST 2016
[+] NSEC: none
[+] NSEC3: none

```

TLD DNS Server

```
[base 10:33] > ./dig -t ns -4 192.168.1.1 www.ubnt.co
;;出入库信息：(192.168.1.1) [192.168.1.1] [www.ubnt.co]
;; 1 server found
;; query: www.ubnt.co ns
;; Got answer:
;; -> QUERY: www.ubnt.co, QUERY: status: NOERROR, id: 23080
;; Flags: qr rd ra; QUERY: 1, ANSWERS: 5, AUTHORITY: 0, ADDITIONAL: 5
;; WARNING: EDNS(0) was requested but not available

; OPT PSEUDOSECTION:
; EDNS: version=8, flags: udp 1452
; ; EDNS(0) options:
; ;   version: 8
; www.ubnt.co.          IN  A
; ; AUTHORITY SECTION:
; www.ubnt.co.          86400  IN  NS  ns2.d-zone.cs.
; www.ubnt.co.          86400  IN  NS  ns1.d-zone.cs.
; ; ADDITIONAL SECTION:
; ns2.d-zone.cs.        86400  IN  A  162.219.64.2
; ns2.d-zone.cs.        86400  IN  AAAA  2a02:3e0:1::2
; ns1.d-zone.cs.        86400  IN  A  162.219.95.2
; ns1.d-zone.cs.        86400  IN  AAAA  2a02:3e0:1::1
; ns1.d-zone.cs.        86400  IN  SRV  1 _http _tcp

; Query time: 0 msec
; SERVER: 192.168.1.1[192.168.1.1]
; WHEN: Wed Feb 10 10:31:02 PST 2028
; MSG SIZE rcvd: 279
```

Authoritative DNS Server

```
[root@fuz2 ~]# dig www.unbo.co  
;;迪 [fuz2 9.18.4s] <> 89.62.239.54.2 www.unbo.co  
;;(No servers found)  
;;  
;; Got answer:  
;;-> SOA unbo.com. unbo. 20200 604800 86400 3600 90  
;; flags: qr rd ra; QUERY: 1, ANSWERS: 1, AUTHORITY: 0, ADDITIONAL: 1  
;; WARNING: recursion requested but not available  
  
;ANSWER SECTION:  
www IN A 192.168.8.72  
OPTOUT IN A 192.168.8.72 60 67 68 39 64 49 73 61 62 65 46 6x (*,recursion disabled)  
www IN A 192.168.8.72  
  
;ANSWER SECTION:  
www IN A 20200 IN DNNAME www.unbo.co.cloudflare.net;  
www IN A 192.168.8.72  
Query time: 99 msec  
SERVING: 162.219.54.143|162.219.54.2  
SOA: 192.168.8.72 192.168.8.72 2020  
MSG SIZE rcvd: 198  
[root@fuz2 ~]
```

Figure 6: Manual iterative DNS query from root server to authoritative name server.

b)

Repeat part a) for the webserver of University of Bonn, i.e. www.uni-bonn.de.

Figure 7: Dig trace for University of Bonn webserver

Figure 7 shows the delegation chain:

- **Root DNS Server:** m.root-servers.net with IP address 202.12.27.33
 - **TLD DNS Server:** z.nic.de with IP address 194.246.96.1
 - **Authoritative DNS Server:** unibn-dns.uni-koeln.de with IP address 134.95.48.166

References

- [1] H. Nielsen et al., *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616, Jun. 1999. DOI: 10.17487/RFC2616. [Online]. Available: <https://www.rfc-editor.org/info/rfc2616>.

- [2] Cloudflare, *What is HTTPS?* <https://www.cloudflare.com/learning/ssl/what-is-https/>, Accessed: 2026-02-11, 2026.