Exercise Session 2: Network Performance - Solutions

COM-208: Computer Networks

Before you start

The goal of this exercise session is to **understand the meaning of throughput and delay**, as **well as the nature of different delay components**. Exercises consist of two types: paper-and-pencil + lab exercises.

Transmission rate and throughput

The **transmission rate** of a link is the **rate at which we can push bits into the link**, and we measure it in "bits per second" (bps), "kilobits per second" (kbps), "megabits per second" (mbps), and so on. Note that 1kbps = 1000bps (not 1024bps), 1mbps = 1 million bps, etc.

The **throughput** between two end-systems is **the rate at which destination receives data**, and we also measure it in "bits per second" (bps), "kilobits per second" (kbps), "megabits per second" (mbps), and so on.

The **maximum throughput** that can be achieved between two end-systems is the **transmission rate of the bottleneck** link between them; i.e, **the link with the slowest transmission rate**.

The different components of delay

A packet experiences different kinds of delay:

- The **transmission delay** experienced by a **packet on a link** is the amount of time it takes to push the bits of the packet onto the link, and it is equal to the packet length divided by the link transmission rate.

- The **propagation delay of a link** is the amount of time it takes for one bit to go from one end of the link to the other, and it is equal to the link length divided by the link propagation speed.
- The **queuing delay** experienced by a packet at a switch is the **amount of time the packet sits inside a queue at the switch**, waiting for other packets to be processed and transmitted. Queuing delay depends on the other traffic that traverses the same links and shares the same queues as the packet.
- The **processing delay** experienced by a packet at a switch is the amount of **time it takes for the switch to process the packet** (after it removes it from the queue and before starting to transmit it). Processing delay depends on the switch's processing capabilities and is typically independent of packet size, at least in most of the scenarios we will discuss in this class.

Delay estimation over a single link

Two end-systems, A and B, are connected by a single link of transmission rate R, length l, and propagation speed c. $R = 100$mbps, $l = 200$km, $c = 2 \cdot 10^8$m/s. What is the propagation delay of the link?

> The propagation delay of the link is $d_{prop} = l/c = 2 \cdot 10^5\text{m}/(2 \cdot 10^8\text{m/s}) = 10^{-3}\text{s} = 1\text{ms}$

A sends two back-to-back packets to B, the first one of size $L_1 = 1$kb, the second one of size $L_2 = 10$kb.

- What is the transmission delay experienced by each packet?

> The transmission delay experienced by the first and second packet, respectively, is $d_{trans,1} = L_1/R = 10^3\text{b}/(10^8\text{b/s}) = 10^{-5}\text{s} = 10\text{us}$. $d_{trans,2} = L_2/R = 10^4\text{b}/(10^8\text{b/s}) = 10^{-4}\text{s} = 100\text{us}$.

- What is the total transfer time, i.e. the time that elapses from the moment A starts transmitting the first bit of the first packet until the moment B receives the last bit of the second packet?

> From Figure 1 we can see that the total transfer time is $d_{prop} + d_{trans,1} + d_{trans,2} = 1.11\text{ms}$

- What is the packet inter-arrival time at B, i.e., the amount of time that elapses from the moment the last bit of the first packet arrives until the moment the last bit of the second packet arrives?

> The packet inter-arrival time at B is equal to the transmission delay experienced by the second packet, i.e. $d_{trans,2} = 100\text{us}$.

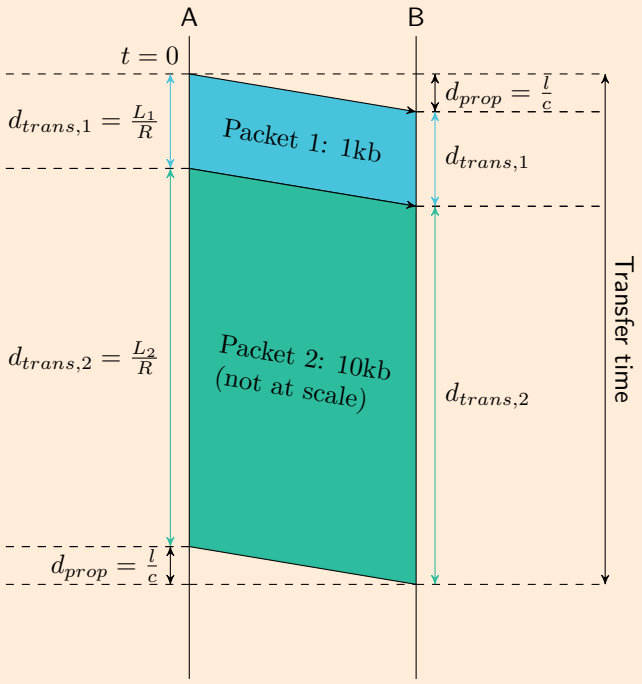

Figure 1: Timing diagram for transmitting two packets in sequence over a single link.

Circuit switching

Let's examine how transfer time changes as we start introducing packet switches between source and destination. Suppose end-systems A and B are connected by not one, but N links and $N-1$ packet switches. All the links have the same properties as the link in the previous problem.

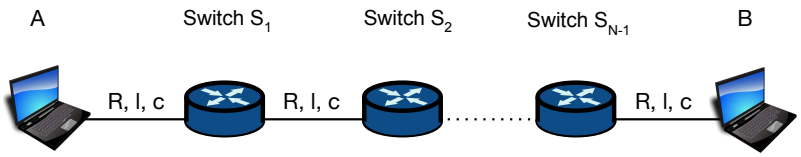

Figure 2: Two end-systems connected through multiple links and packet switches.

Suppose all these packet switches perform *circuit switching*, as in a traditional telephone network: before A starts "talking" to B, a physical circuit is established between them; this enables a packet switch to start transmitting each bit as soon as it receives it. A simple way to think of this scenario is that the packet switches "disappear": if every switch transmits each bit as soon as it receives it, then it is, in a way, as if the switch becomes part of the link, as if A is directly connected to B.

A sends two back-to-back packets to B, each of size $L = 1$kb. Assume zero processing delay.

- What is the total transfer time when N is equal to 2, then 3 links?

> The timing diagram for $N = 2$ links is shown in Figure 3. Due to the fact that the switch uses circuit switching, it does not wait until receiving the full packet; instead it transmits every bit it receives immediately.
>
> The transfer time is $2 \cdot d_{prop} + d_{trans,1} + d_{trans,2} = 2.02$ms.

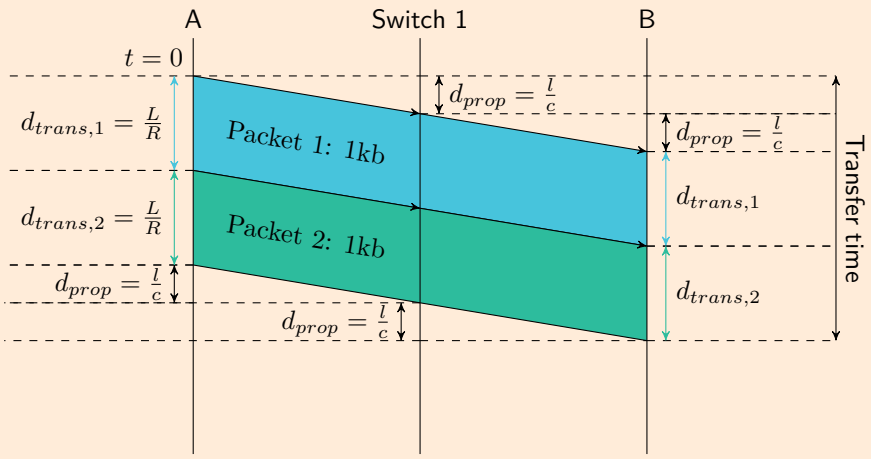

Figure 3: Timing diagram for a transmission over two links connected by a packet switch performing circuit switching.

The timing diagram for $N = 3$ links is shown in Figure 4.

The transfer time is $3 \cdot d_{prop} + d_{trans,1} + d_{trans,2} = 3.02$ms.

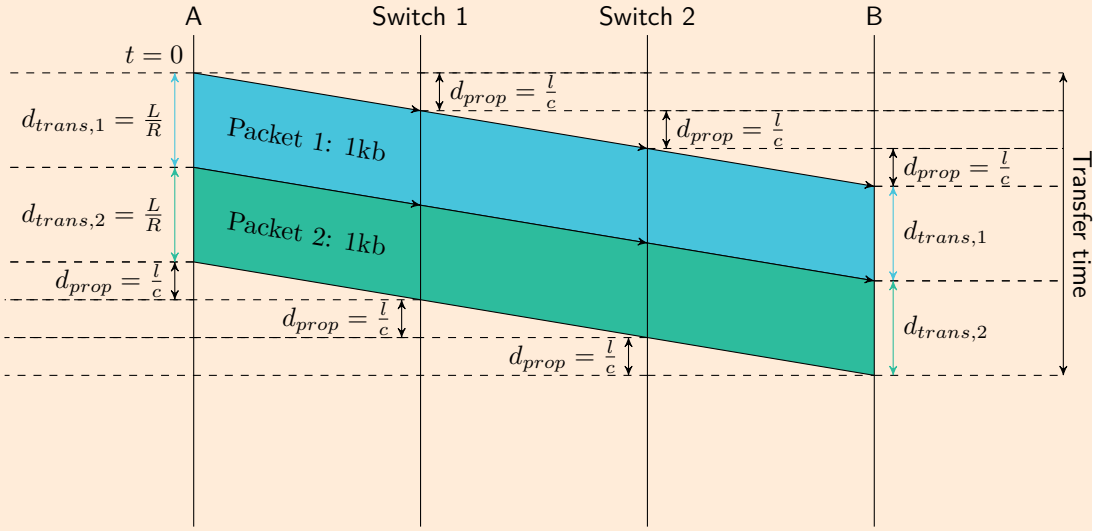

Figure 4: Timing diagram for a transmission over a path with three links, with packet switches performing circuit switching.

- What is the total transfer time as a function of an arbitrary number of links N?

> The circuit behaves like a physical link with rate R and length Nl. The transfer time is $N \cdot d_{prop} + d_{trans,1} + d_{trans,2} = N \cdot l/c + 2 \cdot L/R$

Estimating total delay using timing diagrams

This is a good moment to introduce what we call **timing diagrams**: pictures that represent the various delays that a packet experiences as it travels through a network path. We have started such a diagram right below (Figure 5):

- There is a continuous vertical line for each end-system and packet switch; these lines represent time axes.
- A dashed horizontal line, that crosses the time axes, represents a point in time. For example, on the timing diagram below, the first dashed horizontal line represents $t = 0$, the second one represents $t = d_{prop}$, and so on.
- A continuous line that connects two time axes represents a bit as it travels between the corresponding devices; the beginning of this line represents the moment when the bit was transmitted, while the end of the line represents the moment when the bit arrived at the corresponding device. For example, on the timing diagram below, the first continuous line between the A and Switch1 time axes represents the first bit of the first packet as it travels from A to Switch1; the second continuous line between the same time axes represents the last bit of the first packet.

Using these diagrams, we can easily identify the various delay components that a packet experiences on each link and at each switch. We have marked these for the first packet sent from A to B, when $N = 2$ links. If you add the second packet, you will have solved the problem for $N = 2$. You can draw a new diagram, with 4 time axes, to solve the problem for $N = 3$. You cannot draw a diagram for an arbitrary number of links N, but 2 and 3 links are usually enough to give you the right intuition.

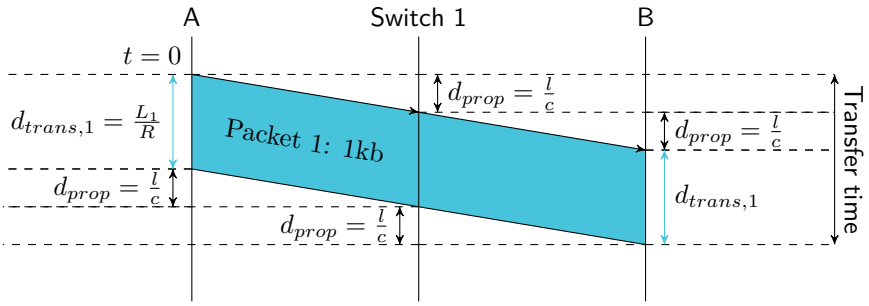

Figure 5: Timing diagram for the transmission of a single packet over two links connected by a packet switch performing circuit switching.

Store and forward

But, Internet packet switches do not typically perform circuit switching; many of them perform store-and-forward: when a switch receives a bit, it must store that bit in a buffer until the last bit of the corresponding packet has arrived; only then can the switch process the packet and start transmitting it over the next link.

Consider the same scenario as in the previous problem, except now, all the packet switches perform store-and-forward. Assume that the processing delay at each switch is 0. Still, the fact that every switch must buffer every bit until it has received the corresponding packet means that there will be some extra delay relative to the circuit switching scenario. (Picture a car rally where every time the first car arrives at an intermediate stop, it must wait for all the cars behind it to arrive before it can start again. Now replace the cars with bits of a packet.)

A sends two back-to-back packets to B, each of size $L = 1$kb.

- What is the total transfer time when N is equal to 2, then 3 links?
- What is the total transfer time as a function of an arbitrary number of links N?

> For this and the above question, the answer is practically the same as below, the only difference being that here $d_{proc} = 0$us.

- How do your answers change when the processing delay at each switch is $d_{proc} = 1$us per packet? Assume that the switch must finish processing and transmitting a packet before it starts processing the next packet.

> The timing diagram for $N = 2$ links is shown in Figure 6. The transfer time is $2 \cdot (d_{prop} + d_{trans,1} + d_{proc}) + d_{trans,2} = 2.032$ms.

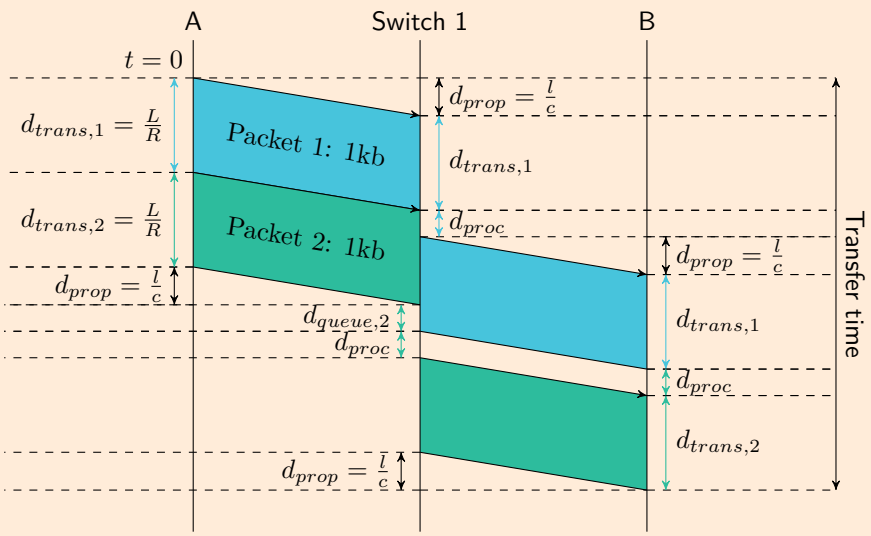

Figure 6: Timing diagram for a transmission over two links connected by a store-and-forward switch.

The timing diagram for $N = 3$ links is shown in Figure 7. The transfer time is $3 \cdot (d_{prop} + d_{trans,1} + d_{proc}) + d_{trans_2} = 3.043$ms.

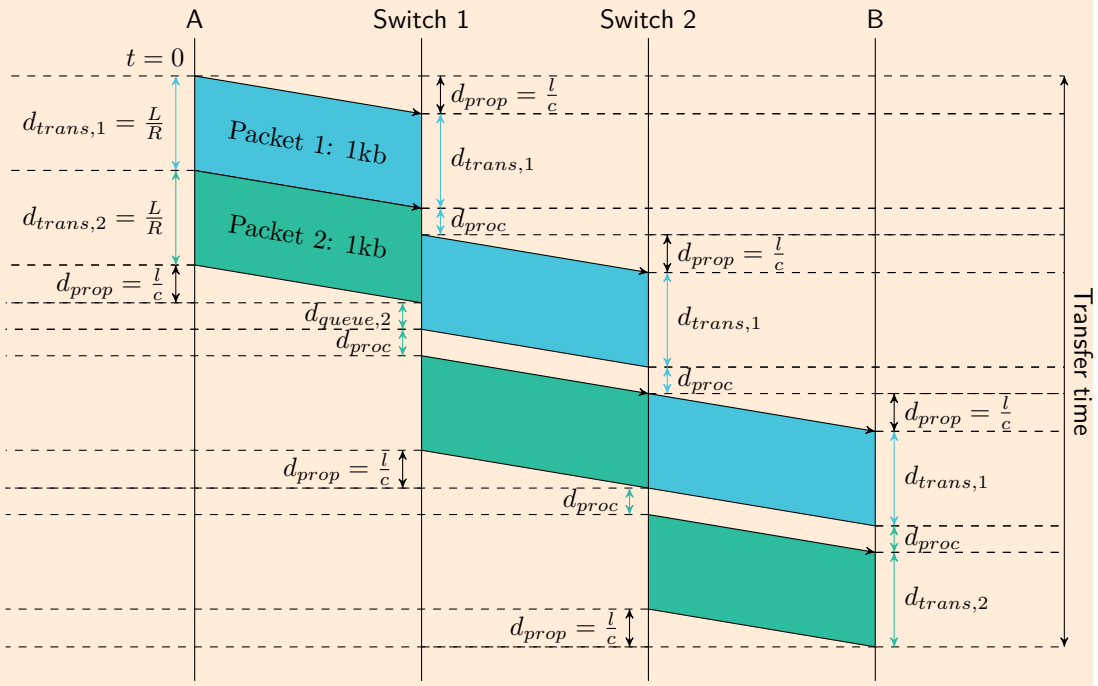

Figure 7: Timing diagram for a transmission over three links connected by store-and-forward switches.

We can see from the previous two cases that as we add one more link, the transfer time increases by $(d_{prop} + d_{trans,1} + d_{proc})$. Thus for $N \geq 2$ links, the transfer time is $N \cdot (d_{prop} + d_{trans,1} + d_{proc}) + d_{trans_2}$.

- Does the second packet experience any queuing delay at the first packet switch? To answer, focus on the first packet switch and compute the difference between: (a) when the switch transmits the last bit of the first packet and (b) when the switch receives the last bit of the second packet. If this is greater than 0, it means that the second packet must wait for the switch to finish transmitting the first packet, i.e., it experiences queuing delay.

Yes, it does. As shown in Figure 6 and Figure 7, at Switch1, the second packet experiences queuing delay $d_{queue,2} = (d_{prop} + d_{trans,1} + d_{proc} + d_{trans,1}) - (d_{prop} + d_{trans,1} + d_{trans,2}) = d_{proc} = 1$us.

- Does the second packet experience any queuing delay at any other packet switch?

> No, because as soon as the last bit of the second packet arrives at any of the subsequent switches, the first packet has already been fully transmitted over the next link.

The easiest way to solve this problem—and most delay problems—is through timing diagrams. We have started one right below that represents the first packet sent from A to B when $N = 2$ links. Notice what is happening at the switch: because it is a store-and-forward switch, it cannot transmit the first bit of the first packet as soon as it receives it, it must wait for the last bit of the first packet to arrive; only then it can process and transmit the first packet.

To solve the problem, add the second packet, then draw another timing diagram for $N = 3$.

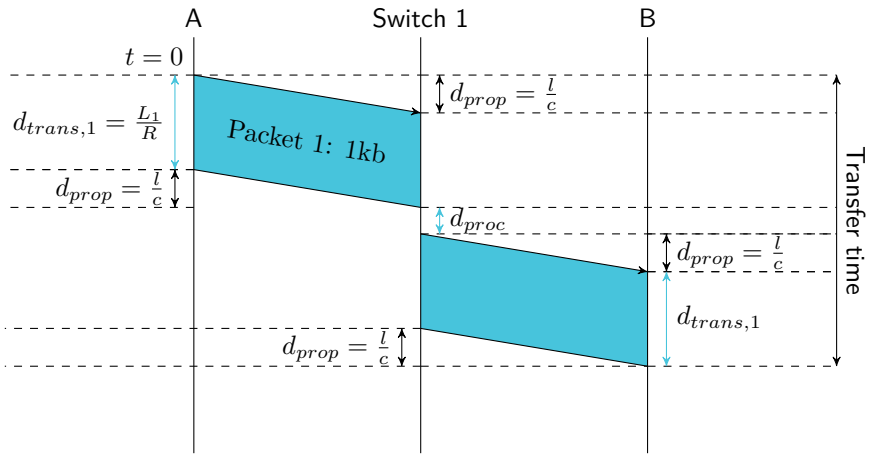

Figure 8: Timing diagram for a transmission over two links connected by a store-and-forward switch.

Now with lots of packets!

When two end-systems communicate over the Internet, they often exchange $P > 2$ packets. Intuitively, that should increase the total transfer time, but by how much? A factor of P? Does it depend on N?

Consider the same scenario as in the previous problem: A and B are connected through $N - 1$ store-and-forward switches, each introducing processing delay 0. Except now, A sends P back-to-back packets to B, all with the same length L. Assume zero propagation delays.

- What is the total transfer time?

Since the timing diagram becomes a bit more complicated when considering multiple links, we may think in the following way:

At time $N \cdot L/R$ the (last bit of the) first packet reaches end-system B, the second packet is ready to be transmitted over the last link, the third packet is ready to be transmitted over the second-to-last link, etc. At time $N \cdot (L/R) + L/R$, the second packet reaches B, the third packet is ready to be transmitted over the last link, etc. Continuing with this logic, we see that all packets will have reached B by time

$$d_{tranfer} = N \cdot L/R + (P-1) \cdot L/R = (N + P - 1) \cdot L/R$$

Queuing and bottlenecks

You may have noticed the artificial uniformity in the previous problems: all packets were of the same length, all links were of the same transmission rate... Reality is not like this, of course. Packets have different lengths, and links have different transmission rates, and both of these things lead to queuing. Even if A and B are alone in the Internet—there is no other traffic to interfere with theirs—when A sends back-to-back packets to B, a packet may have to wait at a packet switch for the previous packet to be transmitted, either because the previous packet is longer, or because the next link is slower.

End-systems A and B are connected through two links with one store-and-forward packet switch in the middle, introducing processing delay 0. Both links have propagation delay d_{prop}. The first link has transmission rate R_1, while the second one has transmission rate R_2.

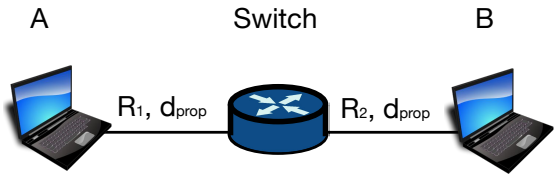

Figure 9: Two end-systems connected through two links.

Assume that $R_1 = R_2$. A sends two back-to-back packets to B, the first one of size L_1, the second one of size $L_2 \neq L_1$.

- In which scenario does the second packet experience queuing delay at the switch? How much?

> In case the first packet is longer than the second packet, i.e., $L_1 > L_2$.
>
> As shown in Figure 10, the queuing delay experienced by the second packet is given by $d_{queue,2} = d_{trans,1} - d_{trans,2}$.

- What is the total transfer time (in this scenario)?

> As shown in Figure 10, the transfer time is $2 \cdot (d_{prop} + d_{trans,1}) + d_{trans,2}$.

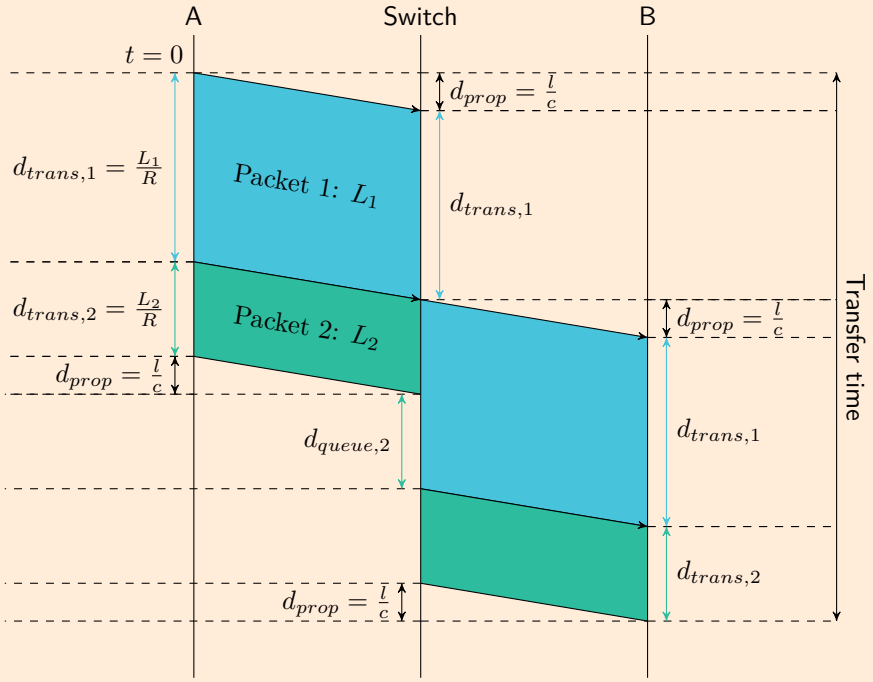

Figure 10: Timing diagram for a transmission over two links connected by a store-and-forward switch.

[This one is if you want to challenge yourself.] A and B are connected though N links and $N-1$ store-and-forward packet switches, each introducing processing delay 0, all links have propagation delay d_{prop} and transmission rate R, and A sends P back-to-back packets to B, of different lengths, $L_1, L_2, ..., L_P$. Assume infinite packet-switch buffers (no packet drops).

- What is the total transfer time?

In general, when the packets have different lengths, the order at which the packets are sent may affect the total transfer time.

However, this is not the case in this exercise where all links have the same processing and propagation delays, the same transmission rate, and packet-switch buffers are infinite (no packet drops). Thus, the total

transfer time is the same as when sending the *longer* packet, say L_1, first and then the rest of the packets. The total transfer time is given by

$$d_{tranfer} = N \cdot \left(\frac{L_1}{R} + d_{prop} \right) + \sum_{i=2}^{P} \frac{L_i}{R}, \quad \text{where} \quad L_1 \geq L_i, \quad \forall i \in [2, 3, \ldots, P]$$

Now let's go back to the simpler scenario where A and B are connected through two links with one store-and-forward packet switch in the middle, introducing processing delay 0. Assume that $R_2 \neq R_1$. A sends two back-to-back packets to B, each of size L. Is it possible that the second packet experiences queuing delay at the packet switch? Let's consider all the possibilities:

- Suppose $R_1 < R_2$, i.e., the first link has a lower transmission rate than the second one, in other words, the first link is the bottleneck. (Picture a narrow local street, followed by a wide highway. Would you expect cars to queue up at the highway entrance?)
  - How much queuing delay does the second packet experience at the packet switch?

    As shown in Figure 11, the second packet experiences 0 queuing delay at the switch.

  - What is the packet inter-arrival time at B?

    As shown in Figure 11, the packet inter-arrival time at B is given by $d_{interarr} = d_{tranfer} - d_{tranfer_1} = d_{trans,2} + d_{trans',2} - d_{trans',1}$.

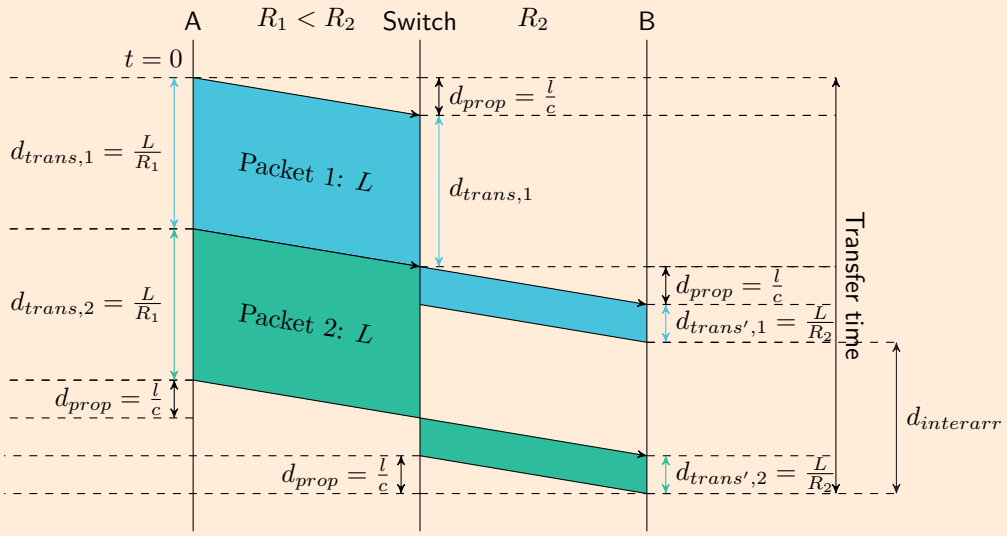

Figure 11: Timing diagram for a transmission over two links connected by a store-and-forward switch.

- Now suppose $R_1 > R_2$, i.e., the second link is the bottleneck. (Picture a wide highway, followed by a narrow local street. Would you expect cars to queue up at the highway exit?)
  - How much queuing delay does the second packet experience at the packet switch?

    As shown in Figure 12, the queuing delay of the second packet at the switch is given by $d_{queue,2} = d_{trans',1} - d_{trans,2}$.

  - Suppose that A sends the second packet T seconds after sending the first one. How large must T be to ensure no queuing at the switch?

    $T \geq d_{queue,2}$ to ensure no queuing delay at the switch.

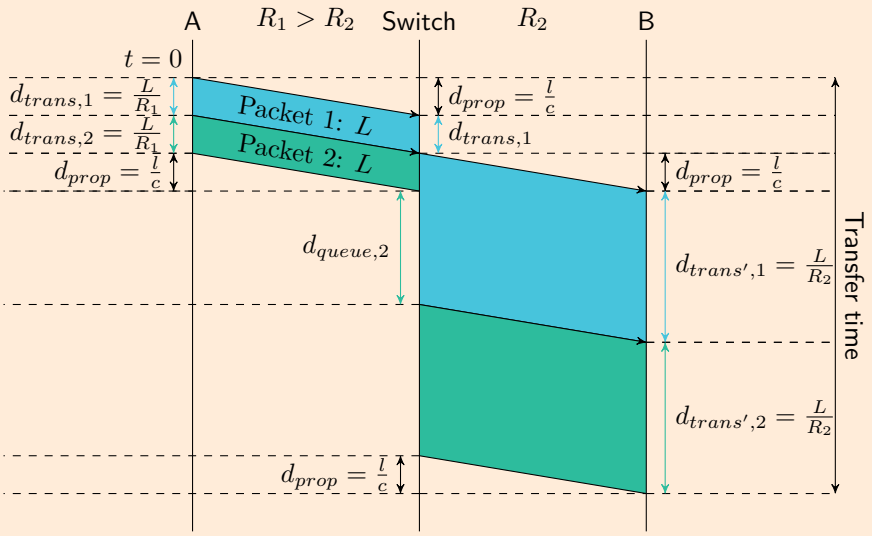

Figure 12: Timing diagram for a transmission over two links connected by a store-and-forward switch.

Lab: Measure delay for different targets

Tools for the lab exercises

For the lab exercises, you will use "ping" (which you discovered in the previous exercise session) to experimentally estimate throughput and delay between your computer and various targets around the world. You will see that there are quantities that we cannot directly measure, yet we can reason about them if we understand what they are.

Download from Moodle `runping.sh` and `plot.sh`. If they are not already executable, make them, e.g., by running `chmod u+x runping.sh` and `chmod u+x plot.sh`. If you are doing the lab on your own computer, make sure you have the `gnuplot` application installed (for Ubuntu `sudo apt install gnuplot`).

`runping.sh` is a shell script that takes as an argument a target (DNS name or IP address) and pings the target many times with different packet sizes. For example, `./runping.sh www.epfl.ch` executes:

```
ping -s 22 -c 50 -i 1 www.epfl.ch > www.epfl.ch-p50
...
ping -s 1472 -c 50 -i 1 www.epfl.ch > www.epfl.ch-p1500
```

The first line pings www.epfl.ch 50 times, once per second, using 50-byte packets, and writes the output to a file called `www.epfl.ch-p50`. The last line pings www.epfl.ch 50 times, once per second, using 1500-byte packets, and writes the output to a file called `www.epfl.ch-p1500`.

`plot.sh` takes as an argument a list of file names (that were produced by 'runping.sh') and produces three new files:

1. `destination_delay.png` shows delay as a function of time, for different packet sizes (different colors correspond to different packet sizes).
2. `destination_scatter.png` shows delay as a function of packet size as a scatter plot.
3. `destination_avg.txt` contains the average (2nd column) and minimum (3rd column) delay values for each packet size (1st column). For example, `./plot.sh www.epfl.ch-p*` produces the above three files for the target www.epfl.ch.

Measure and process

Run `runping.sh` with target www.epfl.ch, then with target www.yahoo.com, and then with target www.gov.kg. Given that you are sending one ping every second, these will take several seconds to complete.

When they finish, process the results by running:

```
./plot.sh www.epfl.ch-p*
./plot.sh www.yahoo.com-p*
./plot.sh www.gov.kg-p*
```

Think about delay

Look at the figures you have produced for each target and answer the following questions:

- Does delay change significantly over time? Whether you answer yes or no, explain why you think this is happening. E.g., if you answered that it does not change, consider each delay component and explain why it doesn't change over time. If you answered that it does change, which delay component(s) are the ones that change?

  Note: If delay has very large spikes, this may be because you are on WiFi. Try connecting to a VDI VM to do the lab.

The following plots (`destination_delay.png`) show delay over time, for different packet sizes (each curve corresponds to a different size):

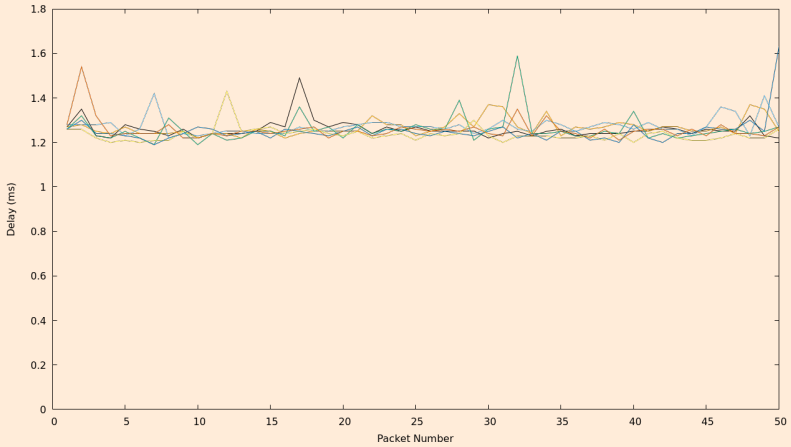

Figure 13: www.epfl.ch

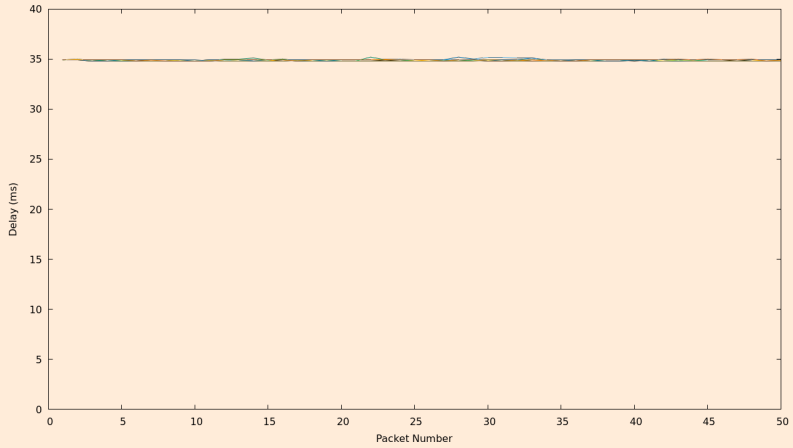

Figure 14: www.yahoo.com

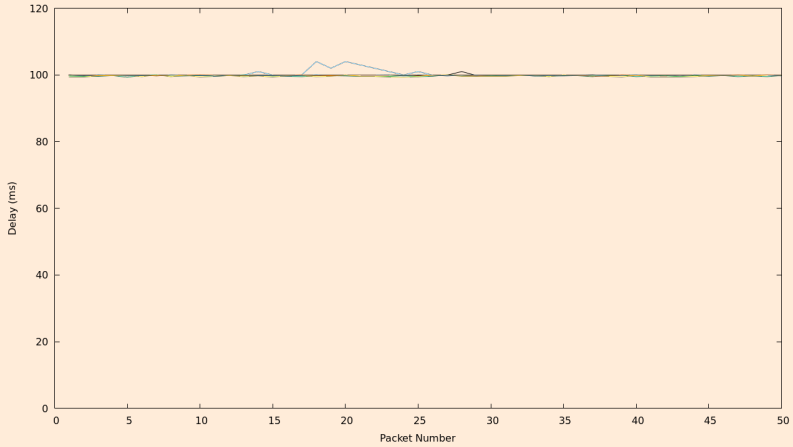

Figure 15: www.gov.kg

Delay does change over time, by hundreds of microseconds. This is most visible in the `www.epfl.ch` graph, because the total delay to this particular target is relatively small (less than 2ms).

When the path between two end-systems remains the same (i.e., consists of the same network links and packet switches), the only delay component that changes significantly over time is the queuing delay, which is affected by cross traffic. So, the change of delay over time that we observe in the graphs could be due to changing queuing delays.

Another possibility is that the path between your computer and each

`ping` target changes over time, in which case it is not only the queuing delay, but the other delay components as well that change. Network-path changes, however, do not typically happen frequently enough to affect an experiment that lasts a few minutes. One exception is load-balancing, where there exist multiple network paths between two end-systems, and each packet is randomly assigned to one of them. However, if this happened between your computer and a `ping` target, you would see it in the delay pattern – packet delays would be clustered around a few distinct values, one for each network path.

Yet another possibility is that it is not (only) the network conditions or paths that change, but the processing delay *at the end-systems*: when a `ping` message arrives at a target, the target takes some time to process it and generate a response; similarly, when a `ping` response arrives at your computer, your computer takes some time to process it and register the time of its arrival. This processing delay may change over time depending on the other processes running on the target and on your computer.

- Do you expect delay to change with the size of the ping packets? Look at the latency vs ping size graph that you produced. Does latency change? How do you explain the result?

The following plots (`destination_scatter.png`) show delay as a function of packet size:

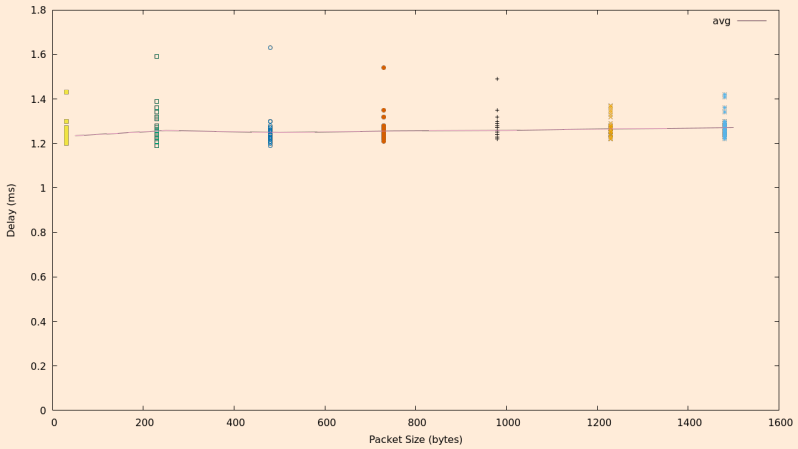

Figure 16: www.epfl.ch

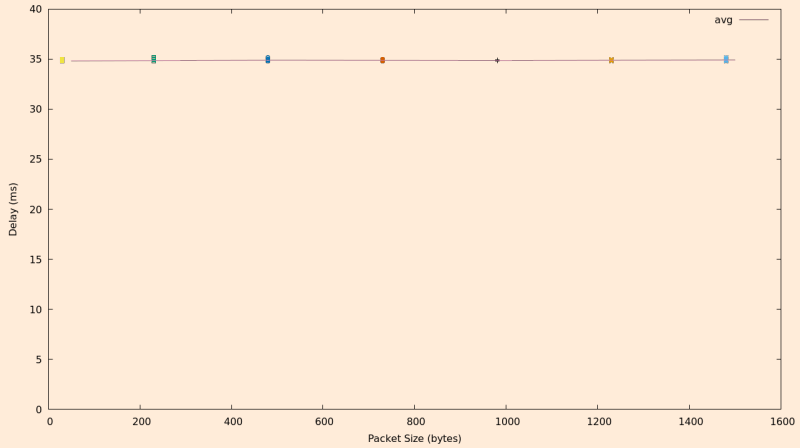

Figure 17: www.yahoo.com

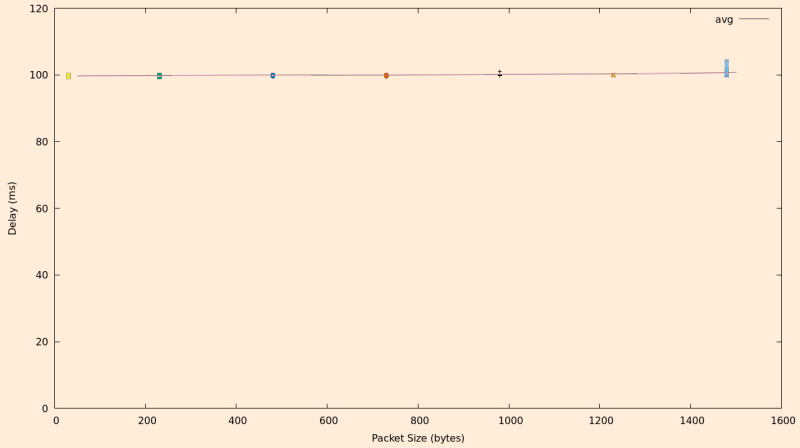

Figure 18: www.gov.kg

In all three plots, it is hard to see that larger packets experience more delay. Generally, we expect larger packets to experience higher transmission delay. As a back-of-the-envelope calculation, assume that the overall transmission rate is 1Gbps (the actual maximum rate of INF3 machines in 2022). The transmission delay for 1400 bytes (the difference between the largest and smallest ping packet) is $1400 * 8/10^9$ seconds or 0.011 milliseconds. This is a very small proportion of the overall latency for all three websites.

Therefore, the reason is that propagation delay (which is independent of packet size) is much higher than transmission delay, so it dominates it.

The above explains why the trend is not *clearly visible* (i.e., why delay does not increase with packet size *significantly*), but it does not explain why, in some cases, the delay experienced by a larger packet may actually be *smaller* than the delay experienced by a smaller packet. The most likely explanation for such an event is that the smaller packet encountered higher queuing delays inside the network, or higher processing delays at the end-systems, and the difference was large enough to counteract its smaller transmission delays.

- Suppose that, for a given target, there is no clear dependence pattern between delay and packet size, i.e., packet size does not affect delay significantly. What does that say about the path from your computer to that target?

That transmission delay on that path is dominated by propagation delay and/or queuing delay.

Estimate delay components

Now we will explore the kind of "rough" estimations that network engineers often need to do.

Ping tells you the overall delay experienced by different packets of different sizes, but it does not (directly) tell you anything about individual delay components, like transmission delay or queuing delay. This is not surprising, because ping operates at the end-systems, it has no access inside the network to see, e.g., how long it takes to transmit each packet on each link, or how long each packet has to wait in a queue at each switch.

Still, if we know the overall delay experienced by *many* different packets and of *many* different sizes, then we *can* infer information about individual delay components.

Study the content of the file `www.yahoo.com_avg.txt`. Estimate the following components of the delay between your computer and www.yahoo.com:

- The propagation+processing delay.
- The average queuing delay.

You don't have enough information to do any precise estimation. Do the best you can given the information you have.

```
$$ cat www.yahoo.com_avg.txt

50 34.825 34.765
250 34.855 34.769
500 34.884 34.809
750 34.872 34.818
1000 34.858 34.800
1250 34.881 34.833
1500 34.900 34.794
```

Propagation+processing delay: A reasonable approximation given the information we have is the minimum delay encountered by any packet sent to the given target, i.e., 34.765ms, in this example. Assuming that the path from your computer to the target and back does not change over time, and that propagation and processing delays do not change over time, we can be certain that the true propagation+processing delay is not larger than our approximation. It is a bit smaller, because even the smallest packet experiences non-zero transmission delays, and it may have experienced non-zero queuing delays; however, by picking the minimum delay encountered by any packet, we minimize this error.

Average queuing delay: A reasonable approximation given the information we have is to compute the difference between the average and minimum delay for each packet size, then compute the average of this difference over all packet sizes. In this example: 1/7 * ((34.825-34.765) + ... + (34.900-34.794)) = 0.07ms. By subtracting the minimum from the average delay for each packet size, we cancel out propagation, processing, and transmission delays. By averaging over all packet sizes, we get as close as possible to the true average queuing delay.

Now study the content of the file `www.gov.kg_avg.txt` and estimate the same delay components between your computer and www.gov.kg. Observe the differences in the results relative to www.yahoo.com. What might explain these differences?

```
$$ cat www.gov.kg_avg.txt

50 99.699 99.372
250 99.761 99.354
500 99.920 99.601
750 100.003 99.676
```

```
1000 100.163 99.826
1250 100.274 99.927
1500 100.730 100.040
```

Using the same approximations:

- Propagation+processing delay: 99.354ms
- Average queuing delay: 1/7 * ((99.699-99.372) + ... + (100.730-100.040)) = 0.39ms

The most notable difference is in the propagation+processing delay. This is not surprising given that this website belongs to the government of Kyrgyzstan, so we expect it to be significantly further away from us than the Yahoo server that responded to our `ping`.

Parallel paths and throughput

Usually, multiple network paths exist between end-systems. Intuitively, adding multiple paths between two end-systems should improve network performance, but which metric exactly? Clearly, adding paths (of the same type) cannot reduce the propagation delay between two end-systems, nor the transmission delay experienced by each single packet. What it *can* change is the overall rate at which A can send data to B: the throughput.

End-systems A and B are connected over M parallel network paths; in this context, "parallel" means that they do not share any links between them. Each path $k \in [1, 2, \ldots, M]$ consists of N links with transmission rates $R_1^k, R_2^k, \ldots, R_N^k$.

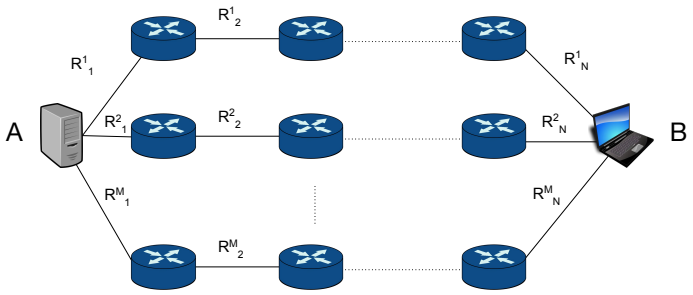

Figure 19: A network topology with multiple parallel paths.

- What is the maximum possible throughput from A to B, when they can use only one path at a time?

> When allowed to use only one path at a time, the maximum possible throughput is the maximum throughput across paths, where for each path k, the throughput is equal to the rate of the slowest link on that path. So overall: $\max_{k=1}^{M} \min(R_1^k, R_2^k, \ldots, R_N^k)$.

- How does your answer change when A and B can use all M paths simultaneously?

> When allowed to use all M paths simultaneously, the maximum possible throughput is the sum of the throughput over each path. So overall: $\sum_{k=1}^{M} \min(R_1^k, R_2^k, \ldots, R_N^k)$.