Computer Systems Tutorial

November 24^{th} 2023

Preliminaries

There were some definitions of bandwidth and delay in the lecture but we will have to be a bit more specific in the exercises.

Bandwidth (digital) It is the data rate of a channel, a quantity measured in bits/sec, i.e. tells how many bits can arrive at a unit of time. That data rate is the end result of using the **analog** bandwidth (measured in Hertz) of a physical channel for digital transmission. Two ways that a digital bandwidth can be increased is either by increasing the analog bandwidth or by sending more than one bit "at once". The formula that relates analog and digital bandwidths is the following (assuming that we are in a noiseless environment)

$$B_{\text{digital}} = 2B_{\text{analog}} log_2 V \text{ bits/sec}$$

where V is the number of discrete levels in the signal (roughly how many bits arrive at once).

Propagation speed s_{prop} This is the speed a signal can travel through a medium, e.g. a wire (Ethernet) or air (WiFi).

Propagation delay This is hence the distance divided by propagation speed. Roughly, it tells how soon a single bit will arrive at the destination (or more than a single bit depending on the encoding).

Bandwidth delay (Transmission delay) It is the time needed for a sender to get the packet onto the wire (note that it is not the time when the receiver receives the packet). This is simply the packet size divided by the bandwidth.

An important difference between bandwidth delay and propagation delay is that bandwidth delay is proportional to the amount of data sent while propagation delay is not. If we send two packets back-to-back, then the bandwidth delay is doubled but the propagation delay counts only once.

Total one-way transmission delay This is hence propagation delay plus bandwidth delay.

There are also queuing delay and **processing** delay that mean the obvious things.

Exercises

Exercise 1 How many edges are in a complete (each node is connected to any other node) graph with 5 nodes? In a graph with n nodes? Assume that only one edge between any two nodes is allowed.

Solution Let's enumerate nodes as a, b, c, d, e. Select node a, we need to connect it to 4 other nodes (no need to connect to itself as we do not allow loops). Then take node b, we do not need to connect it to a as they are already connected, so we have only 3 nodes to connect b to. Then we take c and similarly have only 2 nodes to connect c to. Then we take d and connect it to e. When we take e it is already connected to every node. So, in total, we have $4 + 3 + 2 + 1 = 10$ nodes.

Now consider n nodes. The first one needs to be connected to $n-1$ nodes, the next one to $n-2$ nodes and so on until we get to the penultimate node which needs to be connected only with one other node. This gives us $(n-1)+(n-2)+\ldots+2+1$ nodes. Computing the sum (by using the formula for arithmetic series) we get $\frac{1+(n-1)}{2}(n-1)$ or $\frac{n}{2}(n-1) = \frac{n^2}{2} - \frac{n}{2}$.

Exercise 2

a Consider the following configuration

$$A \longrightarrow B$$

- Propagation delay is 40 µsec
- Bandwidth is 1 byte/µsec (1 mB/sec, 8 Mbit/sec)
- Packet size is 200 bytes (200 µsec bandwidth delay)

What is the total transmission delay to send one packet from A to B?

b Consider the following configuration

$$A \longrightarrow C \longrightarrow B$$

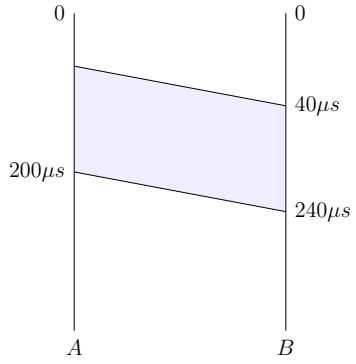
- Propagation delay is 40 µsec (for each wire)
- Bandwidth is 1 byte/µsec (1 mB/sec, 8 Mbit/sec)
- Packet size is 200 bytes (200 µsec bandwidth delay)

What is the total transmission delay to send one packet from A to B?

c The same as above, but with data sent as two 100-byte packets.

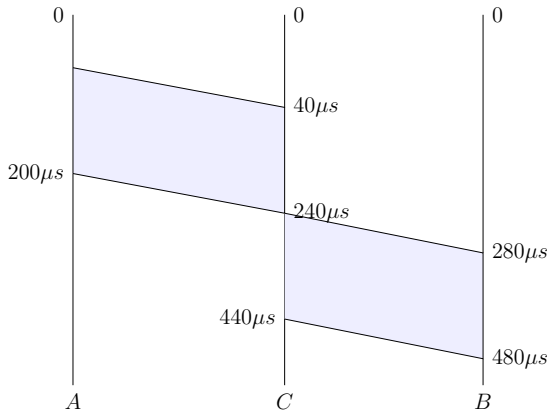
Solution

a Consider the diagram below.



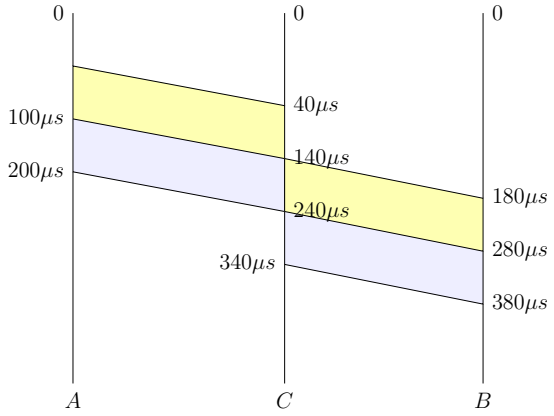
The first bit of the packet needs $40\mu s$ to reach B from A and as the bandwidth is 1 byte/μs we need $200\mu s$ to put the whole packet onto the network wire and the last bit will arrive at time $240\mu s$.

b Consider the diagram below.



The packet arrives from A to C like it did before in part (a). **Because we have store-forward packet-switching, C should wait for the whole packet to arrive before it can send it to B. Once C gets the whole packet, the first bit takes $40\mu s$ to arrive to B and the last bit gets sent at $440\mu s$ and arrives at $480\mu s$. Once again, technically more than one bit can be sent at once but this is encoded inside the bandwidth. Note that at the tutorial I drew the diagram wrong: C sent the bit as soon as it received one. This is not the case in packet-switching.**

c Consider the next diagram



Now we are sending two 100 byte packets instead of one 200 byte packet. So, C wait for the first packet to arrive and sends it to B. The first bit arrives at $40\mu s$ and the last bit of the first packet arrives at $140\mu s$. This is also the time when the first bit of the second packet arrives. Then it waits for the second packet to arrive and sends it to B. Ultimately, the last bit of the second packet arrives at $380\mu s$ to B. **Note that at the tutorial I also drew this one slightly wrong. In my drawing A sent the first bit of the second packet only after the last bit of the first packet arrived at C. This might be the case when C need to send some acknowledgement, but in this particular example it was wrong.**

Exercise 3 Now let us consider the situation when the propagation delay is the most significant component. Suppose we need to send a packet from A to B where the distance between A and B is 10000 km, propagation speed is 200 km/ms. Also, suppose that B has to respond to A with some acknowledgement signal. How many bits of data can we send from A to B before A receives the acknowledgement for the first received bit if the bandwidth is 1 Mbit/sec? 100Mbit/sec?

At most non-LAN scales, the delay is typically simplified to the round-trip time, or RTT: the time between sending a packet and receiving a response. Try executing `ping google.com` in your terminal.

Solution 10000 km with propagation speed of 200 km/ms gives us a propagation delay of 50 ms. Because we also need an acknowledgement from B this results in 100 ms. 1 Mbit/sec is the same as 1 Kbit / ms. Thus, we can manage to send $100 \cdot 1$ Kbit in 100 ms. This is about 12.5 Kbytes. So we can send quite a lot of data before hearing anything back if the propagation delay is significant.

Executing `www.google.com` returns something like `round-trip avg = 22.177 ms` which is the propagation delay back and forth like above.

Exercise 4 This exercise is intended to encourage you to have a look at what various internet protocols look like. Below we will have a look at HTTP. We will need

- Wireshark traffic sniffer `https://www.wireshark.org/`
- Any HTTP server (with no encryption), for example `https://github.com/zowe/sample-node-api` Note that you need `npm` to run the server.

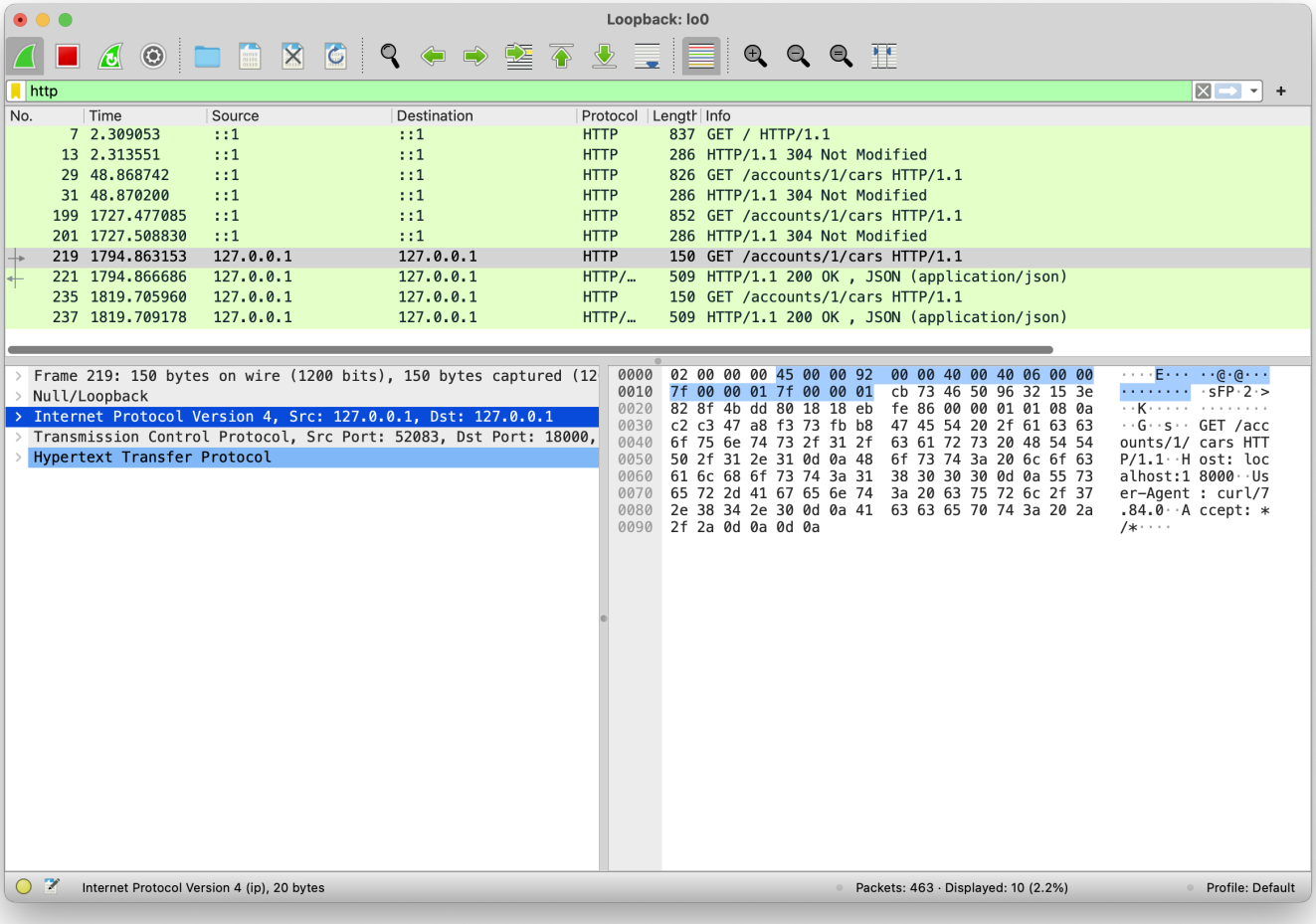


Figure 1: Wireshark example

Clone the repository above, do `npm install;npm start` which will run the server on **localhost:18000**. Start Wireshark and select **loopback** interface for sniffing and start listening on this interface. Then do `curl localhost:18000/accounts/1/cars`. We use `curl` command line utility so that the responses do not get cached. Then stop listening and find the HTTP request that you've sent and HTTP response that you have received. Notice how your HTTP request gets wrapped into TCP and IP packets. The output should look like in Figure 1.

UPD: this will actually work with any web-server. For example, you can make an HTTP request to google by `curl http://www.google.com` and HTTPS request by `curl https://www.google.com`. Compare the corresponding packets in Wireshark (you need to select another interface to listen to, e.g. wifi or ethernet).