CS2105 (Introduction to Computer Networks)

Jia Cheng

Mar 2022

1 Transmission problems

Some variations of transmission problems.

Notation

- Denote A as the source node (sender), B as the destination node (receiver).
- Let $n \geq 1$ be the number of intermediate links, since A, B are distinct. In other words, there are $n-1$ intermediate packet switches.
- Let m be the number of packets.
- Let L be the total size of the data to be sent, so that each packet has size $P := \frac{L}{m}$
- d_{trans} is the transmission time of a single node of a single packet, in the event where transmission times are uniform across all nodes. In the event where the transmission times differ, we can then denote $d_{trans,i}$ as the transmission time of node i. By default, we use 1-indexing, so that $d_{trans,1}$ refers to the transmission time of the sender.
- d_{prop} is the propagation time of a single link, in the event where propagation times are uniform across all nodes.

Transmission delay only In other words, propagation delay, processing delay, queueing delay are ignored.

Let the transmission rate be Rbps

For 1 packet, the total transmission time

$$\text{delay} = n \cdot d_{trans} = n \cdot \frac{P}{R} \tag{1}$$

For all m packets, the total transmission time

$$\text{delay} = (n - 1 + m) \cdot d_{trans} = (n - 1 + m) \cdot \frac{P}{R} \tag{2}$$

To see this, the first $n-1$ is due the time taken for the first packet to finish reaching the $n-1$th packet switch. The remaining m is due to the time taken for the packets to transmit across the last link.

2 links, varying transmission rates There is only 1 packet switch, and A has transmission rate R_1, the packet switch has transmission rate R_2.

For 1 packet, the total transmission time

$$\text{delay} = d_{trans,1} + d_{trans,2} = \frac{P}{R_1} + \frac{P}{R_2} \tag{3}$$

For all m packets, we need to consider cases.

If $R_1 > R_2$ (i.e. R_2 is the bottleneck), then

$$\text{delay} = d_{trans,1} + m \cdot d_{trans,2} = \frac{P}{R_1} + m \cdot \frac{P}{R_2} = \frac{P}{R_1} + \frac{L}{R_2} \tag{4}$$

We can account for the time it takes for the first packet to finish reaching the packet switch. Then, we can simply consider the transmission time of the entire data from the packet switch to B. Intuitively, this is due to the packet switch being slower than A, so we can imagine that when any packet of A initially reaches the switch, they will queue up behind any prior packet.

If $R_1 < R_2$ (i.e. R_1 is the bottleneck), then

$$\text{delay} = m \cdot d_{trans,1} + d_{trans,2} = m \cdot \frac{P}{R_1} + \frac{P}{R_2} = \frac{L}{R_1} + \frac{P}{R_2} \tag{5}$$

The reason for this is that we can imagine that when the last packet reaches the packet switch, there is no queue since R_2 is faster. Hence, this last packet can transmit immediately with time $\frac{P}{R_2}$.

Taking into account propagation delay In some sense, both transmission and propagation are ways of moving signals/electrons/whatnot from one point to another. We can think of (processing and) transmission (heuristically) as moving a packet from one end of the packet switch to the other. Propagation is of course, moving stuff from one end of a link to the other end.

Yet, propagation delay is treated rather differently from transmission delay. The reasons are:

- When transmitting one packet, a packet switch cannot transmit other packets behind in the queue.
- When propagating however, all packets/bits are essentially moving at the same speed. There is no blocking or anything (idealization).

As a result, the end-to-end delay when taking into account propagation delay is

$$\text{delay} = n \cdot d_{prop} + (n - 1 + m) \cdot d_{trans} \tag{6}$$

The derivation of this quantity is as follows:

- Consider the time taken for the first bit of the data to reach the last packet switch. This takes $(n-1) \cdot d_{prop} + (n-1) \cdot d_{trans}$
- At this point, this is actually equivalent to the case of transmitting the data across 1 link. The last bit of the data leaves the last packet switch at time $m \cdot d_{trans}$.
- This last bit then takes d_{prop} to travel across the last link to reach the destination.
- Adding these together, we obtain the desired delay.

Unproven Claim In fact, as a general rule of thumb, the propagation component of the delay is always $\sum_i d_{prop,i}$. The reason is that regardless of how long or short propagation time is, it does not affect the relative "time"-gap between packets.

General problem with varying $d_{trans,i}, d_{prop,i}$ This is also unproven, and I rely on intuition. Not sure how to do a formal proof on this.

The idea is to find the overall bottleneck, i.e. the node with the smallest $d_{trans,i}$. As with the previous paragraph, we can generally ignore propagation delay as the relative time-gap between packets is not affected.

The procedure is then to

- Let $i \in \{1, 2, \ldots, n\}$ be the smallest index that minimizes $d_{trans,i}$.
- Find the time it takes for the first packet to traverse from the sender to node i. Account for propagation delays as well. This gives $\sum_{j<i} d_{trans,j} + \sum_{j<i} d_{prop,j}$.
- Find the time it takes for all the data to be transmitted from node i to node $i+1$. Also account for propagation delay. This gives $m \cdot d_{trans,i} + d_{prop,i}$.

- Find the time it takes for the last packet to traverse from node $i+1$ to the destination. Account for the propagation delays. This gives $\sum_{j>i} d_{trans,j} + \sum_{j>i} d_{prop,j}$.

Adding all these quantities together gives us the total time of

$$\sum_{j \neq i} d_{trans,j} + \sum_{j} d_{prop,j} + m \cdot d_{trans,i} \tag{7}$$

Dividing $\frac{\text{total data}}{\text{total time}}$ gives us the average throughput.

Notice that we could have completely ignored the propagation delay, assuming that it is zero for all links and just add in the term $\sum_i d_{prop,i}$ at the end.

Processing delay This is quite complicated and I'm not too certain about this. We cannot just add processing delay to transmission delay and do something like $n \cdot d_{prop} + (n-1+m) \cdot (d_{proc} + d_{trans})$. The reason is that when a packet is being processed, another packet can be transmitted simultaneously.

2 Utilization problems

Notation

- d_{trans} is the transmission time of a single packet.

Stop-and-wait protocol

$$\text{utilization} = \frac{d_{trans}}{d_{trans} + RTT} = \frac{d_{trans}}{d_{trans} + 2 \cdot d_{prop}} \tag{8}$$

It is true that for a single packet, by the time $d_{trans} + d_{prop}$ has passed, the entirety of the packet has finished reaching the destination. However, we still need to consider the entire RTT since the next packet cannot transmit at this time $d_{trans} + \frac{RTT}{2}$.

Finite number of pipelined packets We disregard any window size, so that all packets can be sent continuously.

Suppose there are m packets.

$$\text{utilization} = \frac{m \cdot d_{trans}}{m \cdot d_{trans} + d_{prop}} \tag{9}$$

The last bit of the data gets out of the sender at time $m \cdot d_{trans}$, and reaches after d_{prop} additional time. Note that at time $m \cdot d_{trans} + d_{prop}$, all the data has been received. Hence, unlike stop and wait, there is no need to consider the whole RTT.

Infinite pipelined stream, finite window size Let n be the window size.

$$\text{utilization} = \min\left\{\frac{n \cdot d_{trans}}{d_{trans} + RTT}, 1\right\} = \min\left\{\frac{n \cdot d_{trans}}{d_{trans} + 2 \cdot d_{prop}}, 1\right\} \tag{10}$$

The numerator is the time taken to transmit everything in the window.
The denominator is the earliest time an ACK from the receiver can reach the sender. The d_{trans} is due to the need to receive the entirety of the first packet before the receiver can process it and generate an ACK. This is also the earliest time the sender can move its window forward, so that the sender can start sending more (new) packets.

Utilization is capped at one, since by definition utilization is the fraction of the time spent transmitting.

3 Cyclic Redundancy Check

Without having taken more advanced algebra courses, I can only make some presumptions about the theory behind CRC32.

Let $P(\mathbb{F})$ denote a polynomial over a field $\mathbb{F}$. Then $P(\mathbb{Z}/2\mathbb{Z})$ denotes the polynomials over $\mathbb{Z}/2\mathbb{Z}$.

For a polynomial $p \in P(\mathbb{Z}/2\mathbb{Z})$, its bitwise representation can be denoted as p_b. Arithmetic then has the following properties.

Addition and subtraction For polynomials p, q,

$$p + q = p - q \tag{11}$$

since $\sum_{i \leq n} a_i x^i = \sum_{i \leq n} -a_i x_i$ for $a_i \in \mathbb{Z}/2\mathbb{Z}$. For instance $1 = -1$ and $0 = -0$.

Also

$$(p + q)_b = p_b \oplus q_b \tag{12}$$

CRC We now give a polynomial arithmetic based derivation for the "checksum" value, where

- D is a degree $d - 1$ polynomial. The d-bit binary data is given by D_b.
- Generator G is a degree r polynomial, i.e. G_b is $r + 1$-bits.
- R is a degree $r - 1$ polynomial, where R_b is the checksum value.

D, G are known, and we wish to calculate R such that $D \cdot x^r + R = P \cdot G$ for some polynomial P.

We manipulate the expression to obtain $D \cdot x^r = P \cdot G - R = P \cdot G + R$, which implies $D \cdot x^r \equiv R(\mod G)$. Hence, R is the remainder polynomial when $D \cdot x^r$ is divided by G.

More explicitly, let's write $D(x), G(x), R(x)$ as the polynomials (instead of implicit function variables D, G, R). Then,

$$\begin{aligned} &D(x)x^r + R(x) \equiv 0(\mod G(x)) \\ &D(x)x^r \equiv -R(x) \equiv R(x)(\mod G(x)) \\ &R(x) = D(x)x^r \mod G(x) \end{aligned} \tag{13}$$