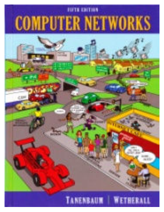
Everything You Ever Wanted to Know About Message Latency



David Wetherall (djw@uw.edu)

1. Message Latency

The *message latency L of sending a message over a link is defined to be the time from which the first bit is sent over the link at the sender until the time at which the last bit is received at the receiver*. Latency is measured in seconds, or more typically milliseconds or even microseconds.

This latency has two components. The first component is the *propagation delay D*, measured in seconds, or t*he time to send a signal from the one end of the link to the other end of the link*. This time depends on the length of the link and the speed at which signals propagate over it. For wireless links, the speed is roughly the speed of light, or 300,000 km/sec. For wired and fiber optic links, it is 200,000 km/sec; signals travel at roughly 2/3 the speed of light in copper and glass.

Often the propagation delay of a link is given directly in seconds, e.g., a 40 ms link, rather than detailing the length of the link, e.g., a 4000 km link across the U.S. This is because it is normally the propagation delay that matters. The propagation delay is often longer than you would expect from the location of the sender and receiver too. This is because, like highways, Internet links do not run directly as the crow flies.

The second latency component is the *transmission delay*, measured in seconds, which is *the time to put the message onto the link at the sender*. It depends on the *size of the message M*, measured in bits, and the *rate of the link R*, measured in bits per second. The rate is a measure of *how many bits are encoded in the signal on the link each second*. The time it takes to encode each bit is therefore the inverse of the rate or $1 / R$. Combining the two factors, the transmission delay is $M \times 1/R$, or M / R. Since the rate of a link is the same at the sender and receiver, the transmission delay is also the time to take the message off the link at the receiver.

Putting both latency components together, we arrive at the expression for the message latency:

$$L = M/R + D$$

To get a feel for message latency, we'll use the analogy of a train going through a tunnel. This setup is shown in Figure 1. The train represents the message; longer trains stand for larger messages. The tunnel represents the link connecting the sender and receiver. The message latency is the time it takes for the train to enter the tunnel and leave the other side. The train track has a speed limit: this represents the data rate. Together, the speed limit and length of the train determine how long it will take the train to enter the tunnel. This time represents the transmission delay. It will grow larger as the train gets longer or the speed limit gets lower. Finally, the length of the tunnel represents the propagation delay. Longer tunnels will take the train longer to cross for a given speed limit and truck length. (To be precise, the length of the tunnel represents the propagation delay *relative* to the transmission delay, where one transmission delay is one train length. So doubling the propagation delay will double the length of the tunnel as you would expect, but only when the rate and length of the train is kept the same.)

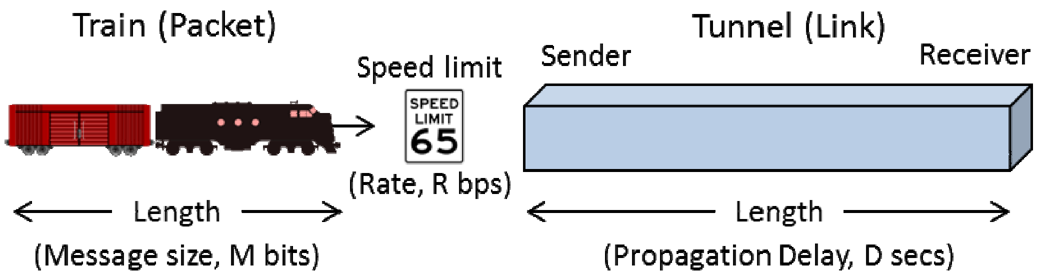


Figure 1: Train analogy for message Latency

We can see the progression of time in Figure 2 for an example train. At *t=0*, the start of the train is poised to enter the tunnel, traveling at the speed limit. At *t=M/R*, the end of the train enters the tunnel so that the train is completely in the tunnel. This is the situation after one transmission delay. At *t=D,* the start of the train leaves the tunnel. This is the situation after one propagation delay. Finally, at *t=M/R* + *D*, the end of the train leaves the tunnel, so that the train has completely cleared the tunnel. This is the situation after the message latency.

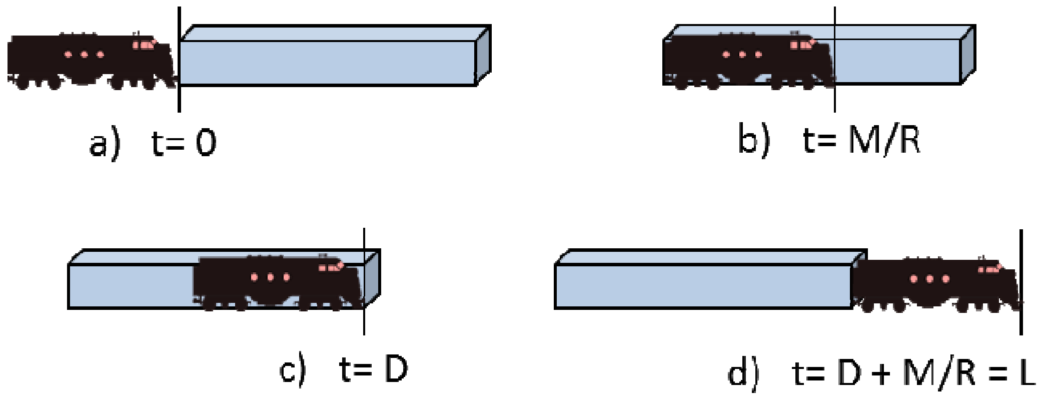


Figure 2: Progression of message latency

Students sometimes wonder why the message latency is not the time to enter the link, plus the time to travel down the link, plus the time to leave the link, i.e., two transmission delays are involved. The answer is parallelism: some bits enter or leave the link while other bits are traveling down the link. For this reason, we need only one propagation delay plus one transmission delay, as we hope the analogy makes clear.

With the link parameters (*D, R*) and message parameters (*M*) it is straightforward to calculate the message latency. There are two different extremes depending on the parameter values that are useful for approximating the latency. For the first extreme, imagine a short train crossing a very long tunnel. This is an exaggeration of Figure 2, in which the tunnel is lengthened. It occurs when *D* >> *M/R*. (Recall that the length of the tunnel of *D* is shown relative to the transmission delay of *M/R*.) In this extreme, by definition, the latency is dominated by the propagation delay; the transmission delay makes little difference. This extreme is the case for "long links" as long as they do not run very slowly.

An example of the first extreme is an ISP cross-country link, say with *D*=30 ms and *R*=10 Mbps. Messages sent over Internet links are usually no more than 1.5 KB. We will consider a 1250 byte or 10,000 bit message for convenience. At 10 Mbps, the message has a transmission delay of 1 ms. So for a link with D=30 ms, the transmission delay makes little difference.

In the second extreme, imagine a short tunnel—so short that most of the train does not fit inside it. We have shown an example in Figure 3. Now the progression of events is different than in Figure 2: the start of the train reaches the far side of the tunnel at *t=D*, well before the end of the train enters the near side of the tunnel at *t=M/R*. Importantly, the expression for message latency holds even in this case—we still need one propagation delay and one transmission delay for the train to clear the tunnel. This extreme occurs when *D* << *M/R* so, by definition, the latency is dominated by the transmission delay; the propagation delay makes little difference. It will be the case for "short links" as long as they do not run at a very high rate.

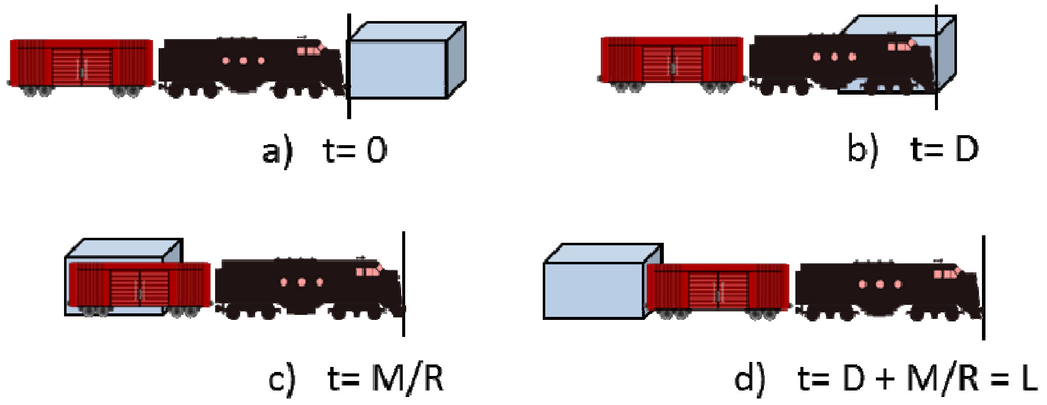


Figure 3: Message latency when D << M/R

An example of the second extreme is an 802.11 wireless link, say running at up to 100 Mbps over a distance of 100 ft. A rule of thumb for the speed of light is that it takes an RF signal 1 ns to propagate 1 ft. So the propagation delay is 100 ns, or 0.1 microseconds. We will again consider a 10,000 bit message for convenience. At 100 Mbps, the transmission delay is 10,000 / 100,000,000 seconds or 0.1 ms. If the rate is less than 100 Mbps the transmission delay will be even larger. While 0.1 ms may appear to be a short time, it is 1000 times larger than the propagation delay. Thus the message latency is roughly the transmission delay of at least 0.1 ms.

The two extremes let us approximate message latency as either the propagation delay for long links, or the transmission delay for short links. With a little experience, you will recognize when each case is applicable. When the propagation delay is a good approximation, you won't need to know message size. You will still need to know the rate to confirm that it is a long link for any reasonable message size. Often this message size is 1500 KB for Internet links, since this is the largest packet that can be sent over a conventional Ethernet link; using it will give an upper bound for the largest transmission delay when the link is used in the Internet. Another complication is wireless links because the data rate may vary depending on the signal-to-noise ratio, and the length of the link may even due to mobility. In the example above we used the highest rate (100 Mbps) as a way to bound the minimum transmission delay and considered a single distance. In different situations, you will need to decide which rate and length are applicable.

2. Latency with Multiple Links

Many situations build on the simple model we have analyzed for the latency of sending a message over a single link. The most important one we will explore is sending a message over multiple links. This happens often because a network path is normally comprised of multiple links.

Let us consider sending a message across multiple links with rates *R1, R2, ...,* and propagation delays *D1, D2, ...* . Each link is joined to the next by a networking device, and the message latency depends on the behavior of thes devices. Normally, each device will completely receive the message from one link before it begins to place the message on the next link. This behavior is called *store-and-forward*. In this case, the message latency over multiple links is simply the sum of the message latency of the individual links:

$$L = M \sum_i 1/R_i + \sum_i D_i$$

It is possible for devices to forward packets without fully receiving them. This behavior is called *cut-through forwarding*. In this case, only the initial transmission delay would be included in the expression above; the links are indistinguishable from one, longer link. However, this behavior is the exception because it is not feasible unless the input and output link run at the same rate. Hence we will concentrate on store-and-forward routing.

For typical Internet paths, we can simplify the expression. Often we will have a measurement of the total propagation delay between sender and receiver. This is $D = \sum_i D_{i.}$. And the slowest rate will often be at the access link at the edge of the network. For example, typical Interent access rates are measured in Mbps, while links within ISP networks and data centers normally run at Gbps. Because of this orders of magnitude difference, the transmission delay of links inside the network is often negligible and can be ignored.

Putting both facts together, we arrive at the observation that an Internet path is often approximated well as a single, longer virtual link. The propagation delay of this link is *D*, the sum of the propagation delays of all links. The transmission delay of this link (if it matters at all since many Internet paths are "long links") is approximated by the transmission delay over the slow, access link with rate *R*. We arrive back at $L = M/R + D$.

3. Round-Trip Time and Bandwidth-Delay Product

Other, important measures of network path build on message latency. The *round-trip time (RTT) is defined as the time for a sender to send a message over a link to a receiver and receive a response back from the receiver*. The response may be a short message, such as an acknowledgement that need only be a single bit long in theory.

The RTT is the sum of two message latencies. We normally assume that the messages use the same link or follow the same network path in different directions with the same propagation delay. Since a typical large message in the Internet may be 1500 bytes and a response may be as short as a single bit, the transmission delay for the response is small enough to ignore. For typical Internet paths, even the transmission delay of the message may be small enough to ignore. With these simplifications we have:

$$RTT = 2D + M/R \approx 2D$$

The RTT is a significant measure because it takes one RTT for actions between the sender and the receiver to take effect. Sending a packet and getting an acknowledgement is one example that takes “an RTT”. For another example, suppose the receiver wants the sender to stop sending traffic. It sends a STOP message at certain time, and after the latency to deliver that message the sender stops. However, the link will still contain data that has already been sent by the sender but is yet to be received by the receiver. The receiver will continue to receive this data for one propagation delay. The total time for the receiver to stop receiving data from the sender after sending the STOP message is one message latency plus one propagation delay. This is equal to one RTT.

The *bandwidth-delay (BD) product* is another important measure of a link or network path. It is defined as *the product of the bandwidth of a link and the delay of the link*. By “bandwidth” we mean the rate of the link or network path. For “delay”, we will use the propagation delay as an approximation. With this simplification we have:

$$BD = R \ x \ D$$

The BD product is a significant measure because it describes the amount of data that is in transit across a link or network path.This is data that has left the sender but has not yet been received by the receiver. Yes, that’s right: links store data. If the bandwidth and the delay are both large then the link can hold a substantial amount of data. For example, consider a cross-country FIOS (fiber Internet access) link. With a rate of 100 Mbps and a delay of 50 ms, up to 5 Mbit of information may be in transit at any given time!

As an example of using the BD-product, suppose we would like to know how much data a receiver needs to be prepared to buffer if it tells a sender to stop sending. Building on our discussion of *RTT* above, the answer is *RTT x R*, which is roughly 2*BD* for long Internet paths.

4. Measuring Latency with `ping`

Turning to practice, you may want to measure the message latency. The most widely used diagnostic tool in all of networking is `ping`, which is intended to determine if a remote host is alive. `ping` sends a message that the remote host echoes. The time between a `ping` request and reply measures the latency of a network path.

What kind of latency have you measured if you "ping" a remote host? `ping` times include all latency components: propagation delay, transmission delay, and queueing delay at routers for all links, plus host processing delays. We have not discussed queueing delay at routers, but because it is variable sending multiple pings and taking the minimum will normally eliminate most of the queuing delay. Host processing delay is kept as small as feasible by implementing `ping` functionality as part of IP, which runs in the operating system. Ignoring these terms, we are left with only the message latency terms for a round-trip. The transmission times may be minimized by sending a short `ping` message. With all these simplifications, `ping` is estimating the round-trip propagation delay of an Internet path. An unavoidable complication of which you should be aware is that Internet paths are frequently asymmetric. That is, the path from B to A is different than the reverse of the path from A to B.

5. To Explore Further

Many elegant experiments build further on the basic notion of message latency in complex networks. A classic example is the design of TCP, which depends on the latency effects of sending multiple messages across a network path. Suppose that the slowest link is in the middle of the network. (This might be the access links, which for many hosts is in the middle because the home or enterprise wireless link is the starting link). If the host sends a small burst of messages as quickly as it can, this burst will be spread out in time by the slow, middle link. The messages will arrive at the receiver preserving this timing even if they now have gaps that were not included when they were first sent! For TCP, the receiver will acknowledge these messages as they come in. The timing gaps between the acknowledgements will also be preserved by the network because the size of an acknowledgment is much, much smaller than the size of a data-carrying packet. So the sender will receive acknowledgments spaced out in time according to the rate of the slowest link. This is an amazing transfer of information about link speeds to the sender! It is used by TCP to pace packets into the network at approximately the bottleneck rate.

To explore further, we suggest that you learn about the operation of `pathchar`. This is a clever tool that infers the bandwidth of the links along a network path by sending probe packets of different lengths, measuring the latency, and solving equations to approximate the rate of successive links (assuming symmetrical network paths). Both TCP congestion control and `pathchar` were designed by Van Jacobson. To learn more, see Allen Downey, "Using `pathchar` to estimate Internet link characteristics," SIGCOMM 1999.

—END—