Calculating Delays

Section 1.4

Delay, Loss, and Throughput in Packet-Switched Networks

Module Goals

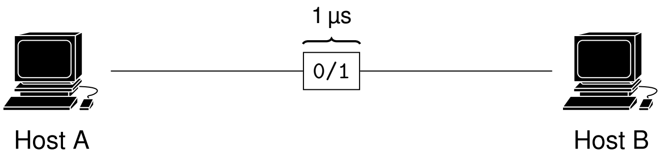
At the conclusion of this module, students will be able to

- explain the difference between computing values that rely on both powers of 2 and powers of 10
- explain the sources of delay in network communications
- calculate delays in various circumstances

Performance

- the predominent metric used to characterize a network's performance is **bandwidth**
  - not a range of frequencies!
  - the number of bits per second that can be transmitted over a communication link
- 1 Mbps $= 10^6$ bits/second $\approx 2^{10}$ bits/second
- 10^{-6} seconds to transmit each bit

  (each bit occupies a 1 µs space on the wire)



Traditional Prefixes

- working with binary objects, we stole perfectly well-defined prefixes and redefined them using powers of 2 instead of powers of 10:
  - Kilo: tranditionally 10^3, now $2^{10} = 1024$
  - Mega: tranditionally 10^6, now $2^{20} = 1048576$
  - Giga: tranditionally 10^9, now $2^{30} = 1073741824$
- using the same prefix for approximately equivalent relationship

Enter Binary Prefixes

- in 1998, the International Electrotechnical Commission (IEC) standardized a suite of units to eliminate this confusion
- the power of 2 prefixes are given new names based on a portmanteau of the SI prefix and the “bi” from binary:
  - Kibi: 2^{10}
  - Mebi: 2^{20}
  - Gibi: 2^{30}
- and the SI prefixes are all powers of 10 again
- **network people always work with the power of 10 prefixes!**

Store and Forward

- packet-switched devices use **store-and-forward** processing
- the device does not receive a bit and immediately retransmit it; instead, every bit of the packet must arrive (and be stored) and only then does it transmit (or forward) the packet
- the device needs to analyze the packet's content to process it
- introduces some delay to the process

Calculating Delay

There are four delays experienced by a packet:

- **propagation delay**: signals travel extremely fast over links, but signals can not travel infinitely fast
- **transmission delay**: bits can only be pushed onto a link at a certain rate, limited by the power of the hardware
- **queuing delay**: packet switches along the way may incur delay because of packets in the outgoing queue
- **proessing delay**: packet switches take some time to determine to which of the outgoing interfaces to route the packet

Calculating Delay

Propagation Delay

- once a bit is pushed into the link, it needs time to propagate to the next network device
- the **propagation time** t_{prop} is simply the distance between the source and destination divided by the **propagation rate** (distance over time)

  (easy to remember with dimensional analysis: ${}^{d}/{}_{d/s} = s$)
- the propogation rate is dependent on the medium:
  - $\approx 2.3 \cdot 10^8$ m/s for copper
  - $\approx 3.0 \cdot 10^8$ m/s for wireless

Calculating Delay

Transmission Delay

- networking hardware can only push bits into a link at a limited rate, or **bandwidth** (its not a truck, but a series of tubes)
  (802.11n @ 600 Mbit/s, Gigabit Ethernet @ 1 Gbit/s, USB3 @ 5 Gbit/s)
- the **transmission time** t_{xmit} is the amount of time it takes to transmit the bits of a packet onto the link
- if a packet of contains L bits of data and is transmitted over a link with a bandwidth R (in bits per second), it takes L/R seconds to transmit the packet
- here, "transmit" only means pushing the packet onto the link; the packet must still propogate to the destination!

Calculating Delay

Queuing Delay

- generally, packets are transmitted in the order that they arrive
- while waiting to be transmitted, the packet is queued and experiences delay
  - if the queue is empty, the packet moves directly to tranmission and experiences no queuing delay
  - if traffic is heavy, the queuing delay may be significant
  - if traffic is too heavy, the queue may fill and a packet might actually be lost
- queuing time is hard to quantify sice it is so traffic dependent, so we'll just call it t_{queue} and specify it when/if it matters

Calculating Delay

Processing Delay

- typically, packets must be introspected to determine where to direct the packet, if there were any bit-level errors that occured during a previous transmission, etc.
- this **processing time** t_{proc} depends strongly on the protocol so we'll ignore it for now
- processing delays on high-speed routers are usually strongly bounded, and are on the order of microseconds

Calculating Delay

The total delay for a single node is then

$$t_{nodal} = t_{proc} + t_{queue} + t_{xmit} + t_{prop}$$

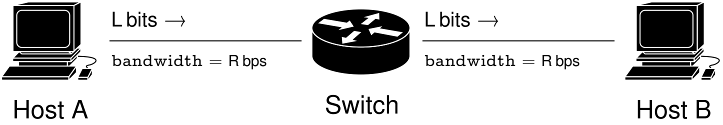
Some implications:

- small transmissions $\rightarrow$ propagation is important
- large transmissions $\rightarrow$ bandwidth is important

Store and Forward Delay

Let's assume:

- there is just one device between the source and destination
- there is just one packet to send
- processing, queuing, and propegation delays are negligable

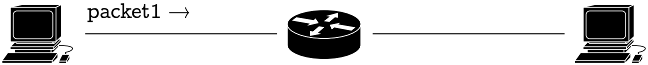


It takes L/R s to send the complete packet to the switch, then L/R s to set the complete packet to the destination.

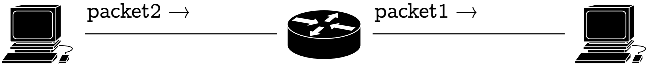
Store and Forward Delay

What if there are multiple packets to send?

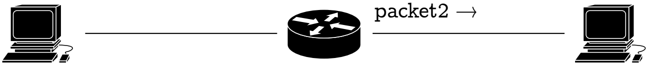
1. (1) sent to the switch ($^{L}/_{R}$ s)



2. (2) sent to the switch, (1) sent to the destination ($^{L}/_{R}$ s)



3. (2) sent to the destination ($^{L}/_{R}$ s)



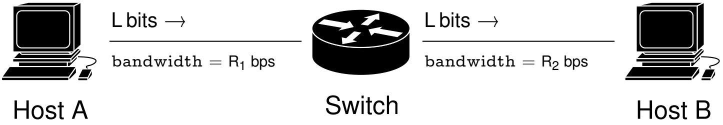
Store and Forward Delay

- what is the end-to-end delay for sending a single packet over a path with N links?

- what is the end-to-end delay for sending a P packets over a path with N links?

Store and Forward Delay

What if links have different transmission rates (bandwidths)?



- $R_1 < R_2$: the second link is limited by the rate of the first link
- $R_1 > R_2$: the second link will get overwhelmed by the first link
- end-to-end transmission rate becomes $\min(R_1, R_2)$

  (assuming a lot of things, e.g., queues are infinitely big if $R_1 > R_2$)

Round-Trip Time (RTT)

- the previous equation was for one-way latency/delay
- most of the time there is a paired request/response—the latency there and back is also important

```
rover-224-252:~ jshiebel$ ping google.com
PING google.com (172.217.5.14): 56 data bytes
64 bytes from 172.217.5.14: icmp_seq=0 ttl=53 time=13.960 ms
64 bytes from 172.217.5.14: icmp_seq=1 ttl=53 time=14.478 ms
64 bytes from 172.217.5.14: icmp_seq=2 ttl=53 time=15.696 ms
64 bytes from 172.217.5.14: icmp_seq=3 ttl=53 time=15.210 ms
64 bytes from 172.217.5.14: icmp_seq=4 ttl=53 time=16.050 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 13.960/15.079/16.050/0.769 ms
```

Link Utilization

Why pay for capability you don't use?

- we should try to use all of of the carrying capacity that a link offers
- if I send data and then wait for a response before continuing, I'm wasting significant capacity!
- instead, we "ask forgiveness rather than permission"
  (good motto for networks, bad motto for life)
- fill the pipe and deal with problems as they come

Link Utilization

- the carrying capacity of a particular link is how much data can be “in flight” between the source and destination
- how much is that?
  - if you want to know how many miles you’ve driven, you multiply the speed of the car by the time you’ve been driving
  - this is the produce of the bandwidth and propegation delay (push bits as fast as you can until the link is “full”)
- but remember that we send data and then are waiting for a response
- that’s two one-way delays, or the round-trip time

Link Utilization

- how much data is "buffered" for a 100 km, 10 Mbps link?
  - propagation delay: $^{100\,\mathrm{km}}/_{2.3 \cdot 10^8\,\mathrm{m/s}} = 0.00043478\,\mathrm{s}$ (0.00086957 s RTT)
  - capacity: $0.00086957\,\mathrm{s} \cdot 10000000\,\mathrm{bps} = 8696\,\mathrm{bits}$ (1087 bytes, or **octets**)
- not a particularly interesting value by iteself
- we'll learn later about the "sliding window protocol" that is heavily reliant on this number
- the capacity, combined with the size of the packet, tells us how many packets we will set before we expect a response from the receiver for the first packet