Csc72010

Parallel and Distributed Computation and Advanced Operating Systems

Lecture 3

February 9, 2006

Network Performance

Notes based on Peterson and Davie, Section1.5

Bandwidth: Number of bits that can be transmitted in a unit of time eg 10Mbps; .1 microsecond per bit; *Bitrate*: 10 bits per microsecond.
You can think of 0.1 microseconds as the "width" of a bit; you can fit 10 bits on the cable every microsecond.
Better technology means "narrower" bits; a 100Mbps NIC puts 100 bits on a cable every microsecond, ie, bits are .01 microseconds wide.

Latency (**delay**, **RTT**=2*delay): How long it takes a signal to travel from one end of the network to the other.

= Propagation + Transmit + Queue
Propagation depends on speed of light through medium (distance/speed)
Transmit depends on Bandwidth (Size of message/bandwidth)
Queuing delays happen when there is congestion in the network.
Bandwidth improves; latency is limited by the speed of light

Factors determining which dominates are file size and distance separating the source and destination:

Small files, large distances?
Large files, small distances?

Instructions per mile:

Suppose100 ms RTT = 5000 miles
1 billion instructions per second: could have executed 100 millon instructions, or 20,000 instructions per mile

Delay * bandwidth product: bits "in flight"

If the sender waits for ACK without filling the pipe, he wastes bandwidth
Mbps versus Gbps cross-country (5000 miles):

	Filesize	RTT
Mbps	1 MB	100 milliseconds
Gbps	1 MB	100 microseconds

Throughput: what can actually be put through wire
Throughput = TransferSize / TransferTime
TransferTime = RTT + 1/Bandwidth * TransferSize

Different kinds of application needs
Data processing: Maximum throughput
Real-time: Max predictability – need to know how bad the jitter is

Computing throughput
Computing throughput, given bytes in a frame and ms of rtt:

1. Convert framesize to bits
2. Use delay or .5*rtt
3. Convert bandwidth to bits per second

If you can send frames continuously:
Throughput in bits per second (bps) = (number of frames *frame size) / total time

Actual throughput = actual bits transferred/time

Filling the pipe:
What happens if you don't send frames continuously? Throughput is reduced.

How much data can a network pipe (connection from source to destination) hold? Since bandwidth is the "width" and latency the "length", it can hold bandwidth*latency bits. Ideally, we always have that many bits "in transit" over a link; it's like having a highway full of cars, that provides the best utilization and the best average throughput (over all users).

Data Link Layer

Notes based on Peterson and Davie, Chapter 2

Nodes

```
CPU
  |              |
Cache            |-----Network Adaptor – (To network)
  |--------------|
Memory
```

CPU speeds increase faster than memory speeds: networking is memory-speed bound!

Links (physical layer)

Speed of light
Full duplex versus half duplex
Cat 5 most common right now – others:
Fiber
Wireless (Bluetooth, 802.11b/g at 2.4 GHz and 802.11a at ?? GHz)
Twisted pair

Cable

Encodings

NRZ: 1 is high signal, 0 is low signal
Problems: baseline wander, recovering clock
NRZI: Transition for a 1, stay at current signal for a 0
Manchester: XOR NRZ-encoded data and the clock, where clock alternates between high and low, with one low-high pair being a cycle.

The main point is that the 1's and 0's are encoded as electrical signals; reading them correctly requires some clock synchronization.

Framing

Data link layer: the data units are called frames (at network layer, called frames). The network adaptor must determine where frames start and end. There is usually a base signal on the line, or there may be noise, so that some distinguishing signal is required.

Byte-oriented

Sentinel approach: start and end with special characters
Problem: special character may appear in data
Solution: escape

Start byte + count

Bit-oriented

Similar to byte-oriented: distinguished bit sequences at beginning & end – use "bit stuffing" to avoid them appearing in the middle of a frame, i.e., 01111110 is the distinguished bit sequence for HDLC. After 5 consecutive 1's in the data, must insert a 0. On receiving side, if bit after 5th 1 is a 0, it was stuffed and is discarded; if it was a 1, then either this is the distinguished bit sequence or there's an error.

Clock-based framing

SONET: Frame is 810 bytes long; there is a special two-byte sequence at beginning. Receiver looks for the two-byte sequence at 810 byte intervals. Synchronizes clock on the two-byte sequence.

Error detection

Two-dimensional parity

One-dimensional: add 1 bit to a 7-bit code (ASCII) or 8-bit (EBCDIC)
Two-dimensional: do it in both directions

Internet Checksum

Used in IP
Add up all transmitted words, and then transmit the sum.
Receiver performs the same computation and compares the result.

Current technique in the Internet: Consider data as a sequence of 16-bit integers. Add them together using 16-bit 1's complement. (-x has every bit of x inverted; when adding, a carryout from the most significant bit is added to the result. Example: -5 and –3:

```
5 0101     -5 1010
3 0011     -3 1100
               0111 (one's complement of 8 = 1000)
```

Weak protection, easy to implement

Cyclic Redundancy Check

Think of a *message as being represented by a polynomial. Coefficient is value of bit. Sender and receiver agree on a divisor polynomial of some degree k. To send a (n+1)-bit message, send* also k additional bits
Multiply M(x) by x^k, that is, add k zeroes to the end of the message. This zero-extended message is T(x).
Divide T(x) by C(x) and find the remainder.
Subtract the remainder from T(x).
Send what is left – this is exactly divisible by C(x).

Facts (page 96 of Peterson and Davie):
CRC can find all single-bit errors, as long as the x^k and x^0 terms of C(x) are non-zero.
CRC can find all double-bit errors, as long as C(x) has a factor with at least three terms.
CRC can find any odd number of errors, as long as C(x) contains the factor (x+1).
CRC can find any "burst" error for which the length of the burst is less than k.

ARQ Protocols

Automatic **R**epeat Re**Q**uest (ARQ)

When frames arrive that can't be corrected – they are dropped. To guarantee reliable FIFO delivery, the sending side must re-send them.

Two pieces to the technique:
Acknowledgment: a control frame that says a data frame has been received correctly
Timeout: Retransmit if there's no acknowledgment before timer goes off or some other event requires re-send.

Three common schemes
– Stop & Wait
– Go Back N
– Selective Repeat
P&D treats the last two as special cases of a single scheme, sliding window.

Stop-and-wait (the original ARQ)

Sender transmits one frame at a time and waits for an ACK
- Receiver ACK's frames
- Sender retransmits frame after a timeout

There's a duplicate problem if the ACK is lost or delayed, because the receiver acknowledged it but the sender re-sends it. So that the receiver can distinguish a re-send from a new frame, attach a 1-bit sequence number alternating between 0 and 1.

Only one frame on a link at a time – way below link capacity.

Remember bandwidth * delay product – we would like to send 8 1-KB frames if the pipe can hold 8 KB.

Hack: Concurrent logical channels in ARPANET
Use stop-and-wait on each of multiple logical channels – so if you can fill a 1-KB pipe with one logical channel using Stop-And-Wait, use 8 logical channels on an 8-KB pipe.

Sliding Window – Data Link Layer Version (Go Back N or SRP)

Go Back N allows the transmission of new frames before earlier ones are acknowledged
Go back N uses a window mechanism where the sender can send frames that are within a "window" (range) of frames
The window advances as acknowledgements for earlier frames are received

Example: Assume full duplex, so frames can be flowing in both directions at once. Then we can send frames 1-8 before we receive the ack for 1; after we get the ack for 1, we can send frame 9.

The N in Go Back N is the number of frames the sender can send without receiving an ACK for the first of the frames. We also call the distance the sender can get ahead of the ACK's the *window size,* because it has to remember all frames after the last acknowledged frame (it may need to re-send them). Once the sender receives the ack for a frame, it can forget it – ie, slide the window past the newly acked frame.

The window starts at the earliest unacknowledged frame (one after the last consecutively acknowledged frame) and contains the next N-1 frames.

Go Back N

Features

Window size = N
frames have a sequence number from 1 to N+1 (use mod N+1 arithmetic)

Sender cannot send frame i+N until it has received the ACK for frame i
- After receiving the ACK for frame i, frames i+1 through i+N are available for sending (th*e window*)

Receiver operates like in Stop and Wait
- Reject out-of-order frames
- Send ACK with number i+1 after receiving all frames up to and including i

Use of piggybacking –
- When traffic is bi-directional ACK is piggybacked on frames going in the other direction
- Each frame in the forward direction contains a SEQ field indicating that frame's sequence number and an ACK field giving the sequence number of the next frame expected.

Example. Sender sends 1,2,4,5,3,7,6,8

Frames 4 and 5 must be rejected by the receiver.
If the sender sends all 8 before receiving an ACK, it starts re-sending with frame 4
If it receives an ACK 2 before sending all 8, meaning it received frame 1 and is looking for frame 2, it can go ahead and send 9

The sender has a "window" of N (or SWS for Sending Window Size – in P&D) frames that can be sent without acknowledgements. This window ranges from the frame last ACKed by the receiver (denoted LAR) to LAR+SWS (note that LAR is one less than the frame number specified in the ACK). When the sender reaches the end of its window, or times out, it goes back and retransmits frame LAR+1. LAR+1 is the smallest number frame not yet ACKed . Let NEXT be the number of the next frame to be accepted from the higher layer (ie, the last frame in the window)

Sender rules.

LAR=-1; NEXT=0 (our window buffer will be 0-based)
• Repeat
– If NEXT <= LAR + SWS (ie, if the entire window has not yet been sent) then send frame NEXT and add 1 to NEXT

– If frame arrives from receiver with ACK > LAR set LAR = ACK-1;

– If LAR < NEXT-1 (there are still some outstanding unacknowledged frames) and sender cannot send any new frames, re-send any frame between LAR+1 and NEXT-1

The last rule says that when you cannot send any new frames you should re-send an old (not yet ACKed) frame
– There may be two reasons for not being able to send a new frame:
Nothing new from higher layer
Window expired (NEXT = LAR + 1 + SWS)
No set rule on which frame to re-send, but least recently sent (earliest) is obvious.

Using P&D notation, LFS (last frame sent) must be a frame with LFS >LAR and LFS <= LAR+SWS, ie, LFS-LAR<=SWS.

Receiver rules.

LFR = -1
• Repeat
– When a good frame arrives, if SEQ=LFR+1 accept frame and add 1 to LFR
At regular intervals send an ACK frame with LFR+1 (next frame expected)
Most DLCs send an ACK whenever they receive a frame from the other direction
Delayed ACK for piggybacking
• Receiver reject all frames with SEQ not equal LFR+1

– However, those frames may still contain useful data

Notes on Go Back N

• Requires no buffering of frames at the receiver
• Sender must buffer up to N frames while waiting for their ACK
• Sender must re-send entire window in the event of an error
• frames can be numbered modulo M where M > N   Because at most N frames can be sent simultaneously
• Receiver can only accept frames in order and Receiver must deliver frames in order to higher layer – This introduces the need to re-send the entire window upon error
• The major problem with Go Back N is this need to re-send the entire window when an error occurs. This is due to the fact that the receiver can only accept frames in order

Selective Repeat Protocol (SRP)

• Selective Repeat attempts to retransmit only those frames that are actually lost (due to errors)   The Receiver must be able to accept frames out of order so it must be able to buffer some frames
• Retransmission requests
Implicit The receiver acknowledges every good frame, frames that are not ACKed before a time-out are assumed lost or in error

Explicit An explicit NAK (selective reject) can request retransmission of just one frame
This approach can expedite the retransmission but is not strictly needed – One or both approaches are used in practice

SRP Rules

• Sender manages window just like GO Back N (Window size SWS)
• Packets are numbered Mod M where M >= 2SWS
• Sender can transmit new packets as long as their number is within W of earliest un-ACKed packets
• Sender retransmit un-ACKed packets after a timeout or on a NAK if NAK is employed
• Receiver ACKs all correct packets
• Receiver stores correct packets until they can be delivered in order to the higher layer

Note the functions that sliding window performs:

1) Guarantees reliable transmission (if window sizes are not too large for sequence numbers)
2) Keeps frames in order

Under some circumstances, it can be used for flow control: throttles the sender because it must wait for ACKs (but not the above algorithm)