Good Ideas So Far…

- Flow control
  - Stop & wait
  - Parallel stop & wait
  - Sliding window
- Loss recovery
  - Timeouts
  - Acknowledgement-driven recovery (selective repeat or cumulative acknowledgement)

Outline

- TCP flow control
- Congestion sources and collapse
- Congestion control basics

Sequence Numbers (reminder)

- How large do sequence numbers need to be?
  - Must be able to detect wrap-around
  - Depends on sender/receiver window size
- E.g.
  - Max seq = 7, send win=recv win=7
  - If pkts 0..6 are sent successfully and all acks lost
    - Receiver expects 7,0..5, sender retransmits old 0..6!!!
  - Max sequence must be $\geq$ send window + recv window
**Sequence Numbers**

- 32 Bits, Unsigned → for bytes not packets!
  - Circular Comparison

- Why So Big?
  - For sliding window, must have $|\text{Sequence Space}| > |\text{Sending Window}| + |\text{Receiving Window}|
  - No problem
  - Also, want to guard against stray packets
    - With IP, packets have maximum lifetime of 120s
    - Sequence number would wrap around in this time at 286MB/s

**TCP Flow Control**

- TCP is a sliding window protocol
  - For window size $n$, can send up to $n$ bytes without receiving an acknowledgement
  - When the data is acknowledged then the window slides forward
  - Each packet advertises a window size
    - Indicates number of bytes the receiver has space for
  - Original TCP always sent entire window
    - Congestion control now limits this

**Window Flow Control: Send Side**

- Packet Sent
  - Source Port
  - Dest. Port
  - Sequence Number
  - Acknowledgment
  - HL/Flags
  - Window
  - D. Checksum
  - Urgent Pointer
  - Options...

- Packet Received
  - Source Port
  - Dest. Port
  - Sequence Number
  - Acknowledgment
  - HL/Flags
  - Window
  - D. Checksum
  - Urgent Pointer
  - Options...

- **App write**
  - acknowledged → sent → to be sent → outside window
Window Flow Control: Receive Side

What should receiver do?

Receive buffer

Aced but not delivered to user

Not yet acked

TCP Persist

• What happens if window is 0?
  • Receiver updates window when application reads data
  • What if this update is lost?
• TCP Persist state
  • Sender periodically sends 1 byte packets
  • Receiver responds with ACK even if it can’t store the packet

Performance Considerations

• The window size can be controlled by receiving application
  • Can change the socket buffer size from a default (e.g. 8Kbytes) to a maximum value (e.g. 64 Kbytes)
  • The window size field in the TCP header limits the window that the receiver can advertise
  • 16 bits $\rightarrow$ 64 KBytes
  • 10 msec RTT $\rightarrow$ 51 Mbit/second
  • 100 msec RTT $\rightarrow$ 5 Mbit/second
  • TCP options to get around 64KB limit $\rightarrow$ increases above limit

Outline

• TCP flow control

• Congestion sources and collapse

• Congestion control basics
Internet Pipes?

- How should you control the faucet?
  - Too fast – sink overflows!
  - Too slow – what happens?

- How should you control the faucet?
  - Too fast – sink overflows!
  - Too slow – what happens?

- Goals
  - Fill the bucket as quickly as possible
  - Avoid overflowing the sink
  - Solution – watch the sink
Plumbers Gone Wild!

- How do we prevent water loss?
- Know the size of the pipes?

Plumbers Gone Wild 2!

- Now what?
- Feedback from the bucket or the funnels?

Congestion

- Different sources compete for resources inside network
- Why is it a problem?
  - Sources are unaware of current state of resource
  - Sources are unaware of each other
- Manifestations:
  - Lost packets (buffer overflow at routers)
  - Long delays (queueing in router buffers)
  - Can result in throughput less than bottleneck link (1.5Mbps for the above topology) \( \Rightarrow \) a.k.a. congestion collapse

Causes & Costs of Congestion

- Four senders – multihop paths
- Timeout/retransmit

Q: What happens as rate increases?
Causes & Costs of Congestion

- When packet dropped, any "upstream transmission capacity used for that packet was wasted!"

Congestion Collapse

- Definition: *Increase in network load results in decrease of useful work done*
- Many possible causes
  - Spurious retransmissions of packets still in flight
  - How can this happen with packet conservation
  - Solution: better timers and TCP congestion control
  - Undelivered packets
  - Packets consume resources and are dropped elsewhere in network
  - Solution: congestion control for ALL traffic

Congestion Control and Avoidance

- A mechanism which:
  - Uses network resources efficiently
  - Preserves fair network resource allocation
  - Prevents or avoids collapse
  - Congestion collapse is not just a theory
  - Has been frequently observed in many networks

Approaches Towards Congestion Control

- Two broad approaches towards congestion control:
  - End-end congestion control:
    - No explicit feedback from network
    - Congestion inferred from end-system observed loss, delay
    - Approach taken by TCP
  - Network-assisted congestion control:
    - Routers provide feedback to end systems
    - Single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
    - Explicit rate sender should send at
    - Problem: makes routers complicated
Example: TCP Congestion Control

- Very simple mechanisms in network
  - FIFO scheduling with shared buffer pool
  - Feedback through packet drops
- TCP interprets packet drops as signs of congestion and slows down
  - This is an assumption: packet drops are not a sign of congestion in all networks
    - E.g. wireless networks
  - Periodically probes the network to check whether more bandwidth has become available.

Outline

- TCP flow control
- Congestion sources and collapse
- Congestion control basics

Objectives

- Simple router behavior
- Distributedness
- Efficiency: \( X = \sum x_i(t) \)
- Fairness: \( (\sum x_i)^2/n(\sum x_i^2) \)
  - What are the important properties of this function?
  - Convergence: control system must be stable

Basic Control Model

- Reduce speed when congestion is perceived
  - How is congestion signaled?
    - Either mark or drop packets
  - How much to reduce?
- Increase speed otherwise
  - Probe for available bandwidth – how?
Linear Control

- Many different possibilities for reaction to congestion and probing
  - Examine simple linear controls
    - Window(t + 1) = a + b Window(t)
    - Different a/b for increase and a_d/b_d for decrease
  - Supports various reaction to signals
    - Increase/decrease additively
    - Increased/decrease multiplicatively
    - Which of the four combinations is optimal?

Phase Plots

- Simple way to visualize behavior of competing connections over time

Phase Plots

- What are desirable properties?
  - What if flows are not equal?

Additive Increase/Decrease

- Both X_1 and X_2 increase/ decrease by the same amount over time
  - Additive increase improves fairness and additive decrease reduces fairness
**Multiplicative Increase/Decrease**

- Both $X_1$ and $X_2$ increase by the same factor over time
- Extension from origin – constant fairness

**Convergence to Efficiency**

**Distributed Convergence to Efficiency**

**Convergence to Fairness**
Convergence to Efficiency & Fairness

- Intersection of valid regions
- For decrease: $a=0$ & $b < 1$

What is the Right Choice?

- Constraints limit us to AIMD
  - Can have multiplicative term in increase (MAIMD)
  - AIMD moves towards optimal point

Important Lessons

- Transport service
  - UDP $\rightarrow$ mostly just IP service
  - TCP $\rightarrow$ congestion controlled, reliable, byte stream
- Types of ARQ protocols
  - Stop-and-wait $\rightarrow$ slow, simple
  - Go-back-n $\rightarrow$ can keep link utilized (except w/ losses)
  - Selective repeat $\rightarrow$ efficient loss recovery
- Sliding window flow control
- TCP flow control
  - Sliding window $\rightarrow$ mapping to packet headers
  - 32bit sequence numbers (bytes)

Important Lessons

- Why is congestion control needed?
- How to evaluate congestion control algorithms?
  - Why is AIMD the right choice for congestion control?
- TCP flow control
  - Sliding window $\rightarrow$ mapping to packet headers
  - 32bit sequence numbers (bytes)
Good Ideas So Far…

- Flow control
  - Stop & wait
  - Parallel stop & wait
  - Sliding window (e.g., advertised windows)
- Loss recovery
  - Timeouts
  - Acknowledgement-driven recovery (selective repeat or cumulative acknowledgement)
- Congestion control
  - AIMD $\rightarrow$ fairness and efficiency

- Next Lecture: How does TCP actually implement these?