Goals of Today’s Lecture

- Finish discussion of Window Scaling
- **TCP Throughput Equation**
  - Computes approximate long-running TCP performance for a given packet loss probability $p$
- Relationship between performance and queuing
  - Router architecture
  - Modeling of queuing systems & Little’s Law
  - FIFO queuing
  - Active Queue Management - RED
  - Explicit Congestion Notification (ECN)

Window Scaling

- Problem: 16 bits of advertised window only allows 65,535 bytes in flight
  - Can significantly restrict throughput
- Solution: TCP window scaling option negotiated in initial SYN exchange
  - $\texttt{shift.cnt}$ specifies scaling factor for units used in window advertisement
  - E.g., $\texttt{shift.cnt} = 5 \Rightarrow$ advertised window is $2^5 = 32$-byte units

Window Scaling, con’t

- Q: **Now** how large can the window be?
- A: Clearly, must not exceed $2^{32}$ …
  - If it does, then can’t disambiguate data in flight
  - So, scaling ≤ 16
- In fact, somewhat subtle requirements limit window to $2^{30}$ to allow receiver to determine whether data fits in the offered window
  - So, scaling ≤ 14

Window Scaling, con’t

- Now we can go fast. Suppose in a high-speed LAN with RTT = 1 msec we can transmit at 1 GB/s.
- What problem arises if packets are occasionally delayed in the network for 10 msec?
- Sequence number wrap: can’t tell earlier, delayed segments from later instances.
- Fix: another TCP option to associate (high-res) timestamps with TCP segments
  - Essentially, adds more bits to sequence space
  - (Side effect: no more need for Karn/Partridge restriction not to compute RTT for ACKs of retransmitted packets)
TCP Throughput Equation

• Consider a network path for which packets are lost with probability $p$ (independently).
• Suppose a TCP connection achieves a maximum window size of $W$ packets when using the path.
  – On AIMD cut due to packet loss, window goes to $W/2$
  – Thus, average window is $w = 3/4 W$
• Total number of packets in the "sawtooth":
  – $W/2 + (W/2 + 1) + (W/2 + 2) + \ldots + W$
  – Sawtooth spans about $W/2$ RTT's
  – Total # packets $= (3/4 W) \cdot (W/2) = 3W^2/8$
• Therefore: $3W^2/8 \approx 1/p$

TCP Throughput Equation, con't

• Given $3W^2/8 \approx 1/p$, then in terms of the average window $w$:
  – $3 \cdot (4/3 w)^2/8 = 1/p$ (since $w = 3/4 W$)
  – $16/9 w^2 = 8/3p \Rightarrow w^2 = 3/2p$
  – $w = \sqrt{3/2} / \sqrt{p}$
• For packets of $B$ bytes, throughput is
  – $T = w \cdot B / RTT = \sqrt{1.5B} / (RTT \cdot \sqrt{p})$
• Implications:
  – Long-term throughput falls as $1/RTT$
  – Long-term throughput falls as $1/\sqrt{p}$
• Non-TCP transport can use equation to provide TCP-friendly congestion control

Where Does Loss (= $p$) Come From, Anyway?

Routers & Queuing

Generic Router Architecture

• Input and output interfaces are connected through an interconnect
• Interconnect can be implemented by
  – Shared memory
    • Low capacity routers (e.g., PC-based routers)
  – Shared bus
    • Medium capacity routers
    • Point-to-point (switched) bus
      • High capacity routers
      • Packets fragmented into cells
      • Essentially a network inside the router!

Shared Memory (1st Generation)

Typically < 0.5Gbps aggregate capacity
Limited by rate of shared memory

Shared Bus (2nd Generation)

Typically < 5Gbps aggregate capacity: Limited by shared bus
Point-to-Point Switch (3rd Generation)

Typically ~ 100Gbps aggregate capacity

What a Router Looks Like

Cisco GSR 12416
Juniper M160

Capacity: 160Gbps
Power: 4.2kW
Lines of Code: 8M (!) (circa year 2000)

Capacity: 80Gbps
Power: 2.6kW

Input Interface

• Packet forwarding: decide to which output interface to forward each packet based on the information in packet header
  – Examine packet header
  – Lookup in forwarding table
  – Update packet header

• Question: do we send the packet to the output interface immediately?

Output Functions

• Buffer management: decide when and which packet to drop
• Scheduler: decide when and which packet to transmit

Input Queued Routers

• Only output interfaces store packets
• Advantages
  – Easy to design algorithms: only one congestion point
• Disadvantages
  – Requires an output speedup (Ro/C) of N, where N is the number of interfaces → not feasible

Output Queued Routers

• Input interfaces store packets
• Easier to build since only need R = C
  – Though need to implement "back pressure" to know when to send
• But harder to build efficiently due to contention and head-of-line blocking
Head-of-line Blocking

- Cell at head of an input queue cannot be transferred, thus blocking the following cells

- Modern high-speed routers use combination of input & output queuing, with flow control & multiple “virtual queues”

Simple Queuing - FIFO and Drop Tail

- Most of today’s routers
- Transmission via FIFO scheduling
  - First-in first-out queue
  - Packets transmitted in the order they arrive
- Buffer management: drop-tail
  - If the queue is full, drop the incoming packet

Queuing Theory

- Enormous literature exists on mathematical modeling of queues
- Abstractions:
  - Arrivals (“customers”) comes to the system according to some probability distribution \( F \) of interarrival times
    - Arrival rate is designated \( \lambda \) (e.g., packets/sec)
  - System has a set of servers (\( \geq 1 \))
  - Each arrival has a service time taken from another probability distribution, \( G \)
- Questions:
  - Given \( F, G, \lambda \) how much delay do customers see?
  - Unfortunately, most solutions require \( F \) with only weak correlations - far from the case for networking

Queuing Example

Packet arrival

\[ d(t) = \frac{Q(t)}{R} + \frac{P}{R} \]

<table>
<thead>
<tr>
<th>Packet arrival</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 Kb</td>
<td>0.5 ms</td>
</tr>
<tr>
<td>1 Kb</td>
<td>1 ms</td>
</tr>
<tr>
<td>1.5 Kb</td>
<td>1.5 ms</td>
</tr>
<tr>
<td>2 Kb</td>
<td>2 ms</td>
</tr>
</tbody>
</table>

\[ Q(t) = \int_{0}^{t} P \, dt \]

Generalized Delay Diagram

Latest bit seen by time \( t \)

Delay (in network/system/router)
**Little's Theorem**

- Assume a system where packets arrive at rate \( \lambda \).
- Let \( d \) be mean delay of packet, i.e., mean time a packet spends in the system.
- \( Q: \) What is \( N \), mean \# of packets in the system?
  - E.g., for a router \( N \) would give the size of the queue.

\[ d = \text{mean delay} \quad \lambda = \text{mean arrival rate} \]

- \( A: \) \( N = \lambda \times d \)

**Little's Theorem: Proof Sketch**

Latest bit seen by time \( t \)

1. Sender
2. Receiver

\[ d(i) = \text{delay of packet } i \]
\[ x(t) = \text{number of bits in transit (in the system) at time } t \]
\[ S = \text{area} \]

What is the system occupancy, i.e., average number of bits in transit between 1 and 2?

\[ S = S(1) + S(2) + \ldots + S(N) \]
\[ = \frac{P(d(1) + d(2) + \ldots + d(N))}{T} \]

Average occupancy = \((\text{average arrival rate}) \times (\text{average delay})\)
Refinements to FIFO

Random Early Detection (RED)
Explicit Congestion Notification (ECN)

Bursty Loss From Drop-Tail Queuing

- TCP depends on packet loss
  - Packet loss is the indication of congestion
  - In fact, TCP drives the network into packet loss
  - ... by continuing to increase the sending rate
- Drop-tail queuing leads to bursty loss
  - When a link becomes congested...
  - ... many arriving packets encounter a full queue
  - And, as a result, many flows perceive congestion
  - ... and in fact tend to become synchronized by near-simultaneous loss

Slow Feedback from Drop Tail

- Feedback comes when buffer is completely full
  - ... even though the buffer has been filling for a while
- Plus, the filling buffer is increasing RTT
  - ... and the variance in the RTT
- Idea: give early feedback
  - Get one or two flows to slow down, not all of them
  - Get these flows to slow down before it is too late
  - Spread out congestion detection to break synchronization

Random Early Detection (RED)

- Basic idea of RED
  - Router notices that the queue is getting backlogged
  - ... and randomly drops arriving packets to signal congestion
- Packet drop probability
  - Drop probability increases with average queue length
  - If buffer is below some level, don’t drop anything
  - ... otherwise, set drop probability as function of length

Properties of RED

- Drops packets before queue is full
  - In the hope of reducing the rates of some flows
- Drops packets in proportion to each flow’s rate
  - High-rate flows have more packets
  - ... and, hence, a higher chance of being selected
- Drops are spaced out in time
  - Helps desynchronize TCP senders
- Tolerant of burstiness in the traffic
  - By basing the decisions on average queue length

RED In Practice

- Hard to get the tunable parameters just right
  - How early to start dropping packets?
  - What slope for the increase in drop probability?
  - What time scale for averaging the queue length?
- RED is implemented in practice
  - E.g., modern routers have config option to turn it on
  - But not clear it’s used (due to tuning challenges)
- Many variations proposed (“Blue”, “FRED”…)
- More generally, think of RED as example of how network can give more refined feedback to end systems regarding current available capacity
  - An example of Active Queue Management (AQM)
Explicit Congestion Notification

- Early dropping of packets
  - Good: gives early/refined feedback
  - Bad: costs a packet drop to give the feedback

- Explicit Congestion Notification (ECN)
  - Router instead marks the packet with an ECN bit
  - ... which end system interprets as a sign of congestion

- Surmounting the challenges
  - Must be supported by both end hosts as well as routers
  - Requires two bits in the IP header (one for ECN mark, one to indicate sender understands ECN)
  - Solution: taken from Type-Of-Service bits in IPv4 packet header
  - Also requires 2 TCP header bits (to echo back to sender)

Summary

- TCP Throughput Equation - if connection not window-limited, then performance scales
  - As 1/RTT
  - As 1/sqrt(p) for loss probability p

- Router architecture: fabric interconnects input interfaces with output interfaces
  - Implications of input-queued vs. output-queued

- Queuing theory & practice
  - Little’s Law relates arrival rate, service time, # in system
  - Simple FIFO queuing predominates
    - RED as exemplar of refined Active Queue Management
    - And ECN as a way to avoid using drops for feedback

- Next lecture: Quality of Service (QOS)