(Transport Layer) Congestion Control
Where we are in the Course

• Still in Transport layer
• (Gently) moving down to include a bit of network layer
TCP to date:

• We can set up a connection (connection establishment)
• Tear down a connection (connection release)
• Keep the sending and receiving buffers from overflowing (flow control)

What’s missing?
Network Congestion

• A “traffic jam” in the network
  • Later we will learn how to control it
Nature of Congestion

• Routers/switches have internal buffering
Nature of Congestion (2)

- Simplified view of per port output queues
  - Typically FIFO (First In First Out), discard when full
Nature of Congestion (3)

• Queues help by absorbing bursts when input > output rate
• But if input > output rate persistently, queue will overflow
  • This is congestion
• Congestion is a function of the traffic patterns – can occur even if every link have the same capacity
Effects of Congestion

• What happens to performance as we increase load?
Effects of Congestion (2)

- What happens to performance as we increase load?
Effects of Congestion (3)

• As offered load rises, congestion occurs as queues begin to fill:
  • Delay and loss rise sharply with more load
  • Throughput falls below load (due to loss)
  • Goodput may fall below throughput (due to spurious retransmissions)

• None of the above is good!
  • Want network performance just before congestion
Bandwidth Allocation

• Important task for network is to allocate its capacity to senders
  • Good allocation is both efficient and fair
• **Efficient** means most capacity is used but there is no congestion
• **Fair** means every sender gets a reasonable share the network
Bandwidth Allocation (2)

• Key observation:
  • In an effective solution, the Transport (end-to-end) and Network (routers) layers must work together

• Network layer witnesses congestion
  • Only it can provide direct feedback

• Transport layer causes congestion
  • Only it can reduce offered load
Bandwidth Allocation (3)

• Why is it hard? (Just split equally among flows!)
  • Number of senders and their offered load changes
  • Senders may lack capacity in different parts of network
  • Network is distributed; no single party has an overall picture of its state
Bandwidth Allocation (4)

• Solution context:
  • Senders (TCP) adapt concurrently based on their own view of the network
  • Design this adaption so the network usage as a whole is efficient and fair
  • Adaption is continuous since offered loads continue to change over time
Fair Allocations
• What’s a “fair” bandwidth allocation?
  • The max-min fair allocation
Recall

• We want a good bandwidth allocation to be both fair and efficient
  • Now we learn what fair means
• Caveat: in practice, efficiency is more important than fairness
Efficiency vs. Fairness

• Cannot always have both!
  • Example network with traffic:
    • A→B, B→C and A→C
  • How much traffic can we carry?
Efficiency vs. Fairness (2)

• If we care about fairness:
  • Give equal bandwidth to each flow
  • A→B: ½ unit, B→C: ½, and A→C, ½
  • Total traffic carried is 1 ½ units
Efficiency vs. Fairness (3)

• If we care about efficiency:
  • Maximize total traffic in network
  • A→B: 1 unit, B→C: 1, and A→C, 0
  • Total traffic rises to 2 units!
The Slippery Notion of Fairness

• Why is “equal per flow” fair anyway?
  • A→C uses more network resources than A→B or B→C
  • Host A sends two flows, B sends one

• Not productive to seek exact fairness
  • More important to avoid starvation
    • A node that cannot use any bandwidth
  • “Equal per flow” is good enough
Generalizing “Equal per Flow”

- **Bottleneck** for a flow of traffic is the link that limits its bandwidth
  - Where congestion occurs for the flow
  - For $A \rightarrow C$, link $A\rightarrow B$ is the bottleneck
Generalizing “Equal per Flow” (2)

• Flows may have different bottlenecks
  • For A→C, link A–B is the bottleneck
  • For B→C, link B–C is the bottleneck
  • Can no longer divide links equally among all flows in a sensible way...

A 1 B 10 C
Max-Min Fairness

• Intuitively, flows bottlenecked on a link get an equal share of that link

• **Max-min fair allocation** is one that:
  • Increasing the rate of one flow will decrease the rate of a smaller flow
  • This “maximizes the minimum” flow
Max-Min Fairness (2)

• To find it given a network, imagine “pouring water into the network”
  1. Start with all flows at rate 0
  2. Increase the flows equally until there is a new bottleneck in the network
  3. Hold fixed the rate of the flows that are bottlenecked
  4. Go to step 2 for any remaining flows
Max-Min Example

• Example: network with 4 flows, links equal bandwidth
Max-Min Example (2)

- When rate=1/3, flows B, C, and D bottleneck R4—R5
  - Fix B, C, and D, continue to increase A
Max-Min Example (3)

• When rate=2/3, flow A bottlenecks R2—R3. Done.
Max-Min Example (4)

- End with $A=2/3$, $B$, $C$, $D=1/3$, and $R_2\rightarrow R_3$, $R_4\rightarrow R_5$ full
Adapting over Time

• Allocation changes as flows start and stop
Adapting over Time (2)

Flow 1 slows when Flow 2 starts

Flow 1 speeds up when Flow 2 stops

Flow 3 limit is elsewhere
Bandwidth Allocation
Recall

• Want to allocate capacity to senders
  • Network (router) layer provides feedback
  • Transport (TCP) layer adjusts offered load
  • A good allocation is efficient and fair

• How should we perform the allocation?
  • Several different possibilities ...
Bandwidth Allocation Models

• Open loop versus closed loop
  • Open: reserve bandwidth before use
  • Closed: use feedback to adjust rates

• Host versus Network support
  • Who is sets/enforces allocations?

• Window versus Rate based
  • How is allocation expressed?

TCP is a closed loop, host-driven, and window-based
Bandwidth Allocation Models (2)

• We’ll look at closed-loop, host-driven, and window-based too

• Network layer returns feedback on current allocation to senders
  • At least tells if there is congestion

• Transport layer adjusts sender’s behavior via window in response
  • How senders adapt is a control law
Additive Increase Multiplicative Decrease

• **AIMD** is a control law hosts can use to reach a good allocation
  • Hosts additively increase rate while network not congested
  • Hosts multiplicatively decrease rate when congested
  • Used by TCP
AIMD

• When things are going well, increase offered load by adding a constant (additive)
  • That is, increase offered load conservatively
  • The increase is the same whether you’re currently using a lot or a little

• When things are going poorly, decrease offered load by multiplying by a constant less than one (multiplicative)
  • That is, aggressively decrease load
  • The more load you’re offering, the larger the amount by which you reduce it
AIMD Fairness / Efficiency

• Flows 1 and 2 share a bottleneck
  • But do not talk to each other directly

• Router provides binary feedback
  • “Tells” flow if network is congested
AIMD Game (2)

• Each point is a possible allocation
AIMD Game (3)

• AI and MD move the allocation

Flow 1

Flow 2

Congested

Fair, y = x

Optimal Allocation

Efficient, x + y = 1

Additive Increase

Multiplicative Decrease
AIMD Game (4)

• Here we go...

![Graph showing AIMD Game]

Flow 1

A starting point

Congested

Fair

Efficient

Flow 2
AIMD Game (5)

• Converges to a good allocation!

A starting point
AIMD Sawtooth

- Produces a “sawtooth” pattern over time for rate of each flow
  - This is the TCP sawtooth
AIMD Properties

• Converges to a set of allocations that are efficient and fair
  • Holds for more general topologies

• Other increase/decrease control laws do not! (Try MIAD, MIMD, MIAD)

• Requires only binary feedback from the network
  • “Try going faster” or “Slow down!”
Feedback Signals

- Several possible signals, with different pros/cons
  - We’ll look at classic TCP that uses packet loss as a signal

<table>
<thead>
<tr>
<th>Signal</th>
<th>Example Protocol</th>
<th>Pros / Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet loss</td>
<td>TCP NewReno Cubic TCP (Linux)</td>
<td>Hard to get wrong</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hear about congestion late</td>
</tr>
<tr>
<td>Packet delay</td>
<td>Compound TCP (Windows)</td>
<td>Hear about congestion early</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Need to infer congestion</td>
</tr>
<tr>
<td>Router indication</td>
<td>TCPs with Explicit Congestion Notification</td>
<td>Hear about congestion early</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Require router support</td>
</tr>
</tbody>
</table>