Lecture 8
Congestion Control

EECS 122
University of California
Berkeley

TOC: Congestion Control
- The Problem
- Questions
- Approaches
- TCP: Algorithm
- TCP Refinements
- Summary

Congestion Control: The Problem
- Flows share links:

How to share the links bandwidth?

Questions: Does it matter?
- Congestion occurs
  - Access link
    - Slow link (56k, DSL, T1, wireless, ...)
  - Access network
    - E.g., behind the DSLAM

Can improve treatment of flows
- E.g., one flow should not get a much smaller fraction of bandwidth
- Some flows might need some guaranteed bandwidth

Questions: Available bandwidth?
- Example:

Example: a link with bandwidth of 3Mbps
(same for 6 and 10)

x, y, z = throughput of flows
Questions: Available bandwidth?

Example:

\[
\begin{array}{ccccccc}
A & 10 & 10 & 3 & 10 & 9 & 10 & B \\
C & 10 & 10 & 3 & 6 & 3 & 10 & F \\
D & 10 & 10 & 3 & 6 & 10 & 3 & 10 \\
E & 10 & 10 & 3 & 6 & 10 & 3 & 10 \\
F & 10 & 10 & 3 & 6 & 10 & 3 & 10 \\
\end{array}
\]

- Assume $C \rightarrow D$ with rate $y$ and $E \rightarrow F$ with rate $z$
- How does $A$ "discover" the available bandwidth to $B$?
- Some approaches:
  1. Reservation
  2. Adapt to congestion
  3. Test for sufficient bandwidth
  4. Pricing congestion

Available bandwidth: Reservation

1. Routers (or manager) keep track of reserved rates
2. $A$ requests a rate $R$ to $B$ from the network
3. The network figures out if $R$ is available
4. If $R$ is available, routers (or manager) update reservations and confirm to $A$
5. Note: Complex, Slow, Requires enforcement, Renegotiations, Pricing

Available bandwidth: Adapt

1. Transmit and slow down if congestion occur
2. Example:
   - Initially: $x = 0$, $y = 3$, $z = 3$
   - Then $A$ increases its rate; $C$ and $E$ notice congestion and slow down
   - Later, $C$ stops; $A$ and $E$ increase rates
3. Notes:
   - No guarantees: throughput may drop
   - Key question: how to adapt rates

Available bandwidth: Test

1. Assume flows require at most 1Mbps (e.g., video)
2. Routers monitor their rates to see if they have at least 1 Mbps of available bandwidth; they mark packets otherwise
3. If $A$ wants a new flow to $B$, it sends test packets to $B$
4. If routers do not mark test packets, then $A$ can start its new flow; otherwise, $A$ does not start it
5. Advantages:
   - 1. relatively simple
   - 2. guarantee

Available bandwidth: Pricing

Example:

\[
\begin{align*}
\text{When they get saturated, routers mark packets} \\
\text{If a flow with rate $R$ uses saturated links, it gets marks with rate $R$} \\
\text{Each mark costs one unit} \\
\text{Source slows down if price becomes excessive} \\
\text{• } x = 1+, y = 2+, z = 2+ \\
\text{→ } pA = 1 + 1; pC = pE = 2 \\
\text{• } x = 2+, y = 1+, z = 1+ \\
\text{→ } pA = 2 + 2; pC = pE = 1
\end{align*}
\]
Questions: What is Fair?

- Example:
  - $x = y = z = 1.5$: fair in max-min sense
  - $x = 0, y = z = 3$: maximizes $x + y + z$
  - $5x = 4y = 4z$: equalizes resources flows
    with $x = 1.33, y = z = 1.67$
  - What if $A \rightarrow B$ needs 2Mbps?
    (and is willing to pay for it)

Congestion Control: Approaches

- Telephone Network: Reservation
- Transmission Control Protocol (TCP)
  - Adapt rate to congestion
  - Algorithm for adaptation attempts to be fair ...
- User Datagram Protocol (UDP)
  - Transmit and hope for the best
- Various proposals for Internet:
  - Reservation
  - Pricing
  - Test
  - Note: Either by hosts or between domains

Congestion Control: TCP Algorithm

- Principles
- Example
- Multiple Sources
- A Bad Algorithm: AIAD
- AIMD: Additive Increase – Multiplicative Decrease
- Why AIAD Fails

TCP Algorithm: Principles

- We focus on the “standard” TCP (reno)
- Idea:
  - Not congested $\Rightarrow$ increase rate
  - Congested $\Rightarrow$ slow down
- Questions:
  - How to detect congestion?
    - Missing ACKs
  - How to increase/slow down?
    - AIMD

TCP Algorithm: Example

- No congestion $\Rightarrow$ $x$ increases by one packet/RTT every RTT
- Congestion $\Rightarrow$ decrease $x$ by factor 2

TCP Algorithm: Multiple Sources

- No congestion $\Rightarrow$ rate increases by one packet/RTT every RTT
- Congestion $\Rightarrow$ decrease rate by factor 2
TCP Algorithm: Bad Algorithm

- **C = 50 pkts/RTT**
- No congestion → x increases by one packet/RTT every RTT
- Congestion → decrease x by 1

TCP Algorithm: AIMD

- Limit rates: x = y

TCP Algorithm: Why AIMD Fails

- Limit rates: x and y depend on initial values

Congestion Control: TCP Refinements

- **Fast Retransmit**
- **Fast Recovery**: 1st Look
- **Fast Recovery**: 2nd Look
- **Slow Start**
- **Window Updates**
- **Flow Control**
- **Summary**

Refinements: Fast Retransmit

- Cumulative ACKs:
  - ACK # = next expected #
- 3rd duplicated ACK:
  - likely packet loss
  - retransmit

Refinements: Fast Recovery (1)

- Timeout → Reset Window = 1 unit (MSS)
- 3rd Dup ACK → Window/2

- Slope = 1 MSS/RTT
- n/2
- n
- 1

Moderate congestion

Severe congestion
**Refinements**: Fast Recovery (2)

- Window adjustment is tricky:
  - Want $W \rightarrow W/2$
  - $ssthresh = W/2$ when get new ACK
  - $W = ssthresh + 3$
  - $W = W + 1$ at each Dup Ack

$$W = n + W/2$$

**Refinements**: Slow Start

- Objective: Discover available bw fast
- Solution: Exponential increase of window

$$65KB$$

**Refinements**: Window Updates

- Exponential: $W = W + 1$ at each ACK:
  - $W = 1$
  - $W = 2$
  - $W = 4$
  - $W = 8$

- Additive: $W = W + 1/W$ at each ACK:
  - $W = 8.125 + 1/8.125$
  - $W = 8 + 1/8$
  - $W = 8 + 8/8 = 9$
  - $W = 9 + 9/9 = 10$

**Refinements**: Flow Control

- Objective: Avoid saturating destination
- Algorithm: Receiver advertises window RAW
  - Window = min{RAW – OUT, W}

**Refinements**: Summary

- Actual window = min{RAW - OUT, W}

**Congestion Control**: Summary

- Slow Start: Discover available bandwidth
- Congestion Avoidance: AIMD \(\Rightarrow\) Tries to be fair
- Refinements:
  - Fast Retransmit: 3DA
  - Fast Recovery: Reset $W$ to $W/2$ (instead of $W = 1$)
    - More precisely: $ssthresh = W/2$, $W = W + 1$ per DA,
    - $W = ssthresh$ when get new ACK.
  - TO: set $ssthresh = W/2$, $W = 1$, SS until $W = ssthresh$, then CA

- Timers:
  - Timeout = Average + 4 Deviations
  - If timeout \(\Rightarrow\) Timeout x 2 Reset after new packet or new ACK

- Flow Control:
  - Window = min{RAW – OUT, W}