CS434/534: Topics in Network Systems

Reliable, High-Performance, Secure Network Systems Transport: Rate/Congestion Control

Oct. 7, 2021

Y. Richard Yang
Computer Science Department, Yale University
Email: yry@cs.yale.edu

http://zoo.cs.yale.edu/classes/cs434/
Programming project 1 due this Friday

- Office hours before project 1 due
  - Thursdays 8:00-9:00 pm (instructor; AKW 208A)
  - Fridays 11:00-12:00 pm (instructor; AKW 208A)
  - Fridays 3:00-4:00 pm (Jian's zoom)
  - Fridays 4:00-5:30 pm (instructor; AKW 208A)

- Install ab tool on zoo requested but not installed yet, use your own/friend's machine

Class planning:

- Oct. 12: hacking day
- Oct. 19: G2 guest lecture
Recap: Sliding Window Transport Structure and TCP Instantiation

- Sliding window TCP instantiation
  - Bi-directional transport, accum ack + sack options, delayed send, delayed ack, timeout, fast retransmit

- Connection management and TCP instantiation
  - Connection setup
    - The authentication principle
      - Challenge/response
      - TCP TWH (three-way-handshake)
  - Connection close
    - The no-finite-message close problem
    - Timeout based close
      - TCP 4 way close
TCP FSM

%netstat -t -a

Example see:
http://lxr.linux.no/linux+v4.15.14/net/ipv4/tcp_input.c#L4003
Recap: Why Rate/Congestion Control

When sources sending rate too high for the network to handle:

- Packet loss =>
  - wasted upstream bandwidth when a pkt is discarded at downstream
  - wasted bandwidth due to retransmission (a pkt goes through a link multiple times)

- High delay
Recap: Basic Rate Control Structures

- Explicit rate control \( x(t) \)
  - Explicit pacing of packets

- Implicit rate control, e.g.,
  - Window control as a form of rate control

\[
\text{Rate} = \frac{cwnd \times MSS}{RTT} \text{ Bytes/sec}
\]
Outline

- Admin and recap
- Reliable, High-Performance, Secure Network Transport
  - Overview
  - Basic reliable transport structures
  - Basic transport (congestion) rate control
    - Why rate/congestion control?
    - Basic rate control design
The Desired Properties of a Rate Control Scheme

- Efficiency: close to full utilization but low delay
  - fast convergence after disturbance

- Fairness (resource sharing)

- Scalable implementation (e.g., totally distributed rate control)
Simple Model for Distributed Rate Control Design

Flows observe congestion signal $d$, and locally take actions to adjust rates.
Linear Control

- Proposed by Chiu and Jain (1988)
- Considers the simplest class of control strategy

\[
x_i(t+1) = \begin{cases} 
   a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\
   a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}
\]
State Space of Two Flows

\[ x_i(t+1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
  a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} 
\end{cases} \]
congestion

\[
x_i(t+1) = \begin{cases} 
    a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
    a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}
\]
Implication: Congestion (overload) Case

- In order to get closer to efficiency and fairness after each update, decreasing of rate must be \textit{multiplicative decrease} (MD)
  - $a_D = 0$
  - $b_D < 1$

$$x_i(t + 1) = \begin{cases} a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\ b_D x_i(t) & \text{if } d(t) = \text{cong.} \end{cases}$$
no-congestion

\[ x_i(t+1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
  a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases} \]
Implication: No Congestion Case

- In order to get closer to efficiency and fairness after each update, additive and multiplicative increasing (AMI), i.e.,
  - $a_I > 0$, $b_I > 1$

\[
x_i(t+1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
  b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}
\]

- Simply additive increase gives better improvement in fairness (i.e., getting closer to the fairness line)
- Multiplicative increase may grow faster
Build Intuition: State Transition Trace of AIMD Control

- **Fairness Line:** $x_1 = x_2$
- **Efficiency Line:** $x_1 + x_2 = C$

The diagram shows the transition trace of AIMD control, with points indicating states $x_0$, $x_1$, and $x_2$. The graph illustrates the dynamics between underload and overload conditions.
Exercise: State Trace Analysis of Four Special Cases

<table>
<thead>
<tr>
<th></th>
<th>Additive Decrease</th>
<th>Multiplicative Decrease</th>
</tr>
</thead>
<tbody>
<tr>
<td>Additive Increase</td>
<td>AIAD (b_I=b_D=1)</td>
<td>AIMD (b_I=1, a_D=0)</td>
</tr>
<tr>
<td>Multiplicative Increase</td>
<td>MIAD (a_I=0, b_I&gt;1, b_D=1)</td>
<td>MIMD (a_I=a_D=0)</td>
</tr>
</tbody>
</table>

\[
x_i(t + 1) = \begin{cases} 
  a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\
  a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.} 
\end{cases}
\]
Basic idea: identify a metric (potential function), and consider its change

- AIMD as an example:
  - define $m = |x_1 - x_2|$  
    - During AI, $m = \text{does not change}$  
    - During MD, $m = m/2$

Exercise:
- Why MIMD does not converge?  
- Why MIAD diverge?
Outline

- Admin and recap
- Reliable, High-Performance, Secure Network Transport
  - Overview
  - Basic reliable transport structures
  - Basic transport (congestion) rate control
    - Why rate control?
    - Basic rate control design
      - Basic AIMD model
      - Implementation (mapping AIMD model to implementation): TCP/Reno
Mapping Abstract A(M)I-MD to Protocol

Basic questions to look at:
- How to obtain $d(t)$ -- the congestion signal?
- What values do we choose for the formula?
- How to map formula to code?

$$x_i(t+1) = \begin{cases} 
  a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\
  b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}$$
Obtain $d(t)$ Approach 1: End Hosts Consider Loss as Congestion

Pros and Cons of endhosts using loss as congestion?

Assume loss $\Rightarrow$ cong
Obtain $d(t)$: Approach 2: Network Feedback (ECN: Explicit Congestion Notification)

Sender 1

Sender 2

Receiver

Sender reduces rate if ECN received.

Receiver bounces marker back to sender in ACK msg

Network marks ECN Mark (1 bit) on pkt according to local condition, e.g., queue length > K

Pros and Cons of ECN?
Mapping Abstract A(M)I-MD to Protocol

Basic questions to look at:
- How to obtain $d(t)$--the congestion signal?
- What values do we choose for the formula?
- How to map formula to code?

$$x_i(t+1) = \begin{cases} 
  a_I + b_I x_i(t) & \text{if } d(t) = \text{no cong.} \\
  b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}$$
TCP/Reno Formulas

- **Multiplicative Increase (MI)**
  - double the rate: \( x(t+1) = 2 \times x(t) \)

- **Additive Increase (AI)**
  - Linear increase the rate: \( x(t+1) = x(t) + 1 \)

- **Multiplicative decrease (MD)**
  - half the rate: \( x(t+1) = \frac{1}{2} \times x(t) \)
TCP/Reno Formula Switching (Control Structure)

- **Two “phases”**
  - **slow-start**
    - Goal: getting to equilibrium gradually but quickly, to get a rough estimate of the optimal of $cwnd$
    - Formula: $MI$
  - **congestion avoidance**
    - Goal: Maintains equilibrium and reacts around equilibrium
    - Formula: $AI \cdot MD$
TCP/Reno Formula Switching
(Control Structure)

- Important variables:
  - cwnd: congestion window size
  - ssthresh: threshold between the slow-start phase and the congestion avoidance phase

- If cwnd < ssthresh
  - MI

- Else
  - AIMD
Mapping Model (Formula) to Implementation

- Model (synchronous): increase by **1 unit** per **round**

\[ x_i(t + 1) = \begin{cases} 
  a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
  b_D x_i(t) & \text{if } d(t) = \text{cong.} 
\end{cases} \]

- Exercise: what is reasonable unit, what is reasonable round?
  - 1 unit typically means one MSS (minimal segment size, e.g., 512 bytes)
  - one round typically means one round trip time

- Exercises:
  - Time function mapping; cwnd in unit of MSS
    - \( cwnd(t_0) = w_0 \); \( cwnd(t_0+\text{RTT}) = w_0+1 \); \( cwnd(t) = w_0+(t-t_0)/\text{RTT} \); \( t \) in \([t_0, t_0+\text{RTT}]\)
  - Time function mapping; cwnd in unit of bytes
    - \( cwnd(t_0) = w_0 \); \( cwnd(t_0+\text{RTT}) = w_0+\text{MSS} \); \( cwnd(t) = w_0+(t-t_0)\times\text{MSS}/\text{RTT} \); \( t \) in \([t_0, t_0+\text{RTT}]\)
  - Per ack update, cwnd in unit of bytes
    - Assume send window full, cwnd increases to \( cwnd + \text{MSS} \) after getting all cwnd bytes acked
    - \( cwnd += \text{bytes}_\text{acked}/cwnd \times \text{MSS} \)
MI: Slow-start in Code

Initially:

\[
cwnd = 1;
\]
\[
ssthresh = \text{infinite (e.g., 64K)};
\]

For each newly ACKed segment:

\[
\text{if (cwnd < ssthresh)}
\]
\[
\text{/* MI: slow start*/}
\]
\[
cwnd = cwnd + 1;
\]
TCP/Reno Full Alg in Code

Initially:
  cwnd = 1;
  ssthresh = infinite (e.g., 64K);

For each newly ACKed segment:
  if (cwnd < ssthresh) // slow start: MI
     cwnd = cwnd + 1;
  else
     cwnd += 1/cwnd; // congestion avoidance; AI

Triple-duplicate ACKs:
      // MD
     cwnd = ssthresh = cwnd/2;

Timeout:
  ssthresh = cwnd/2; // reset
  cwnd = 1;
  (if already timed out, double timeout value; this is called exponential backoff)

Many details configured in OS kernel
TCP/Reno: Big Picture

TD: Triple duplicate acknowledgements
TO: Timeout
Outline

- Admin and recap
- Reliable, High-Performance, Secure Network Transport
  - Overview
  - Basic reliable transport structures
  - Basic transport (congestion) rate control
    - Why rate control?
    - Basic rate control design (AIMD, TCP/Reno)
    - Basic rate control analysis
Objective

- To understand the throughput of TCP/Reno as a function of RTT (RTT), loss rate (p) and packet size
- The understanding is a base for many modern designs such as TCP Cubic, MPTCP, BBR, ...

- We will analyze TCP/Reno under two different settings
TCP/Reno Throughput Analysis Common Setting

- mean packet loss rate: $p$; mean round-trip time: $RTT$, packet size: $S$
- Consider only the congestion avoidance mode (long flows such as large files)
- Assume no timeout
- Assume mean window size is $W_m$ segments, each with $S$ bytes sent in one RTT:

$$Throughput = \frac{W_m \times S}{RTT} \text{ bytes/sec}$$
Outline

- Admin and recap
- Reliable, High-Performance, Secure Network Transport
  - Overview
  - Basic reliable transport structures
  - Basic transport rate (congestion) control
    - Why rate control?
    - Basic rate control design (AIMD, TCP/Reno)
    - Basic rate control analysis
      - small fish in a big pond (*high multiplexing* setting)
TCP/Reno Throughput Modeling (High-Multiplexing)

\[ \Delta W = \begin{cases} 
\text{if the packet is not lost} \\
\text{if packet is lost} 
\end{cases} \]

\[
\text{mean of } \Delta W = 0
\]

\[
\Rightarrow \text{mean of } W = \sqrt{\frac{2(1-p)}{p}} \approx \frac{1.4}{\sqrt{p}}, \text{ when } p \text{ is small}
\]

\[
\Rightarrow \text{throughput } \approx \frac{1.4S}{RTT\sqrt{p}}, \text{ when } p \text{ is small}
\]

This is called the TCP throughput sqrt of loss rate law.
Outline

- Admin and recap
- Reliable, High-Performance, Secure Network Transport
  - Overview
  - Basic reliable transport structures
  - Basic transport (congestion) rate control
    - Why rate control?
    - Basic rate control design (AIMD, TCP/Reno)
    - Basic rate control analysis
      - small fish in a big pond (*high multiplexing* setting; fixed loss rate)
      - big fish in a small pond (*low multiplexing* setting)
TCP/Reno Throughput Modeling: Relating $W$ with Loss Rate $p$

Total packets sent per cycle = \((W/2 + W)/2 \times W/2 = 3W^2/8\)

Assume one loss per cycle \(\Rightarrow p = 1/(3W^2/8) = 8/(3W^2)\)

\[\Rightarrow W = \sqrt{\frac{8}{3p}} = \frac{1.6}{\sqrt{p}}\]

\[\Rightarrow \text{throughput} = \frac{S}{RTT} \cdot \frac{3}{4} \cdot \frac{1.6}{\sqrt{p}} = \frac{1.2S}{RTT \sqrt{p}}\]
Exercise

- Assume
  - MSS = 1KB
  - 2 long TCP reno flows with RTT=10ms compete on the bottleneck
  - bottleneck BW = 200M

- What is the loss rate of each TCP flow?

  \[
  \frac{1.2 \times \text{MSS}}{\text{RTT} \times \sqrt{p}} = T
  \]

  \[
  \Rightarrow p = \left(\frac{1.2 \times \text{MSS}}{\text{RTT} \times T}\right)^2
  \]

  \[
  \Rightarrow p = \left(\frac{1.2 \times 10^3}{0.01 \times 10^8}\right)^2 = 0.0001
  \]
Exercise

- What are the transmission rates of $x_1$, $x_2$, and $x_3$?

$$x_1 = \frac{1}{2\sqrt{2}} x_2$$

 Rates: 
- $x_1 = \frac{1}{1+2\sqrt{2}} = 0.26$
- $x_2 = x_3 = 0.74$

$$\Rightarrow \text{throughput} = \frac{S}{RTT} \frac{3}{4} \frac{1.6}{\sqrt{p}} = \frac{1.2S}{RTT \sqrt{p}}$$
A Puzzle: cwnd and Rate of a TCP Session

Q1: cwnd fluctuates widely (i.e., cut to half); how can the sending rate stay relatively smooth?
TCP/Reno Queueing Dynamics

If the buffer at the bottleneck is large enough, the buffer is never empty (not idle), during the cut-to-half to “grow-back” process, smoothing the sending rate – this is called rate matching using buffers.

Offline Exercise: How big should the buffer be to achieve full utilization?