Network Transport Layer:
TCP/Reno Analysis, TCP Cubic, TCP/Vegas

Y. Richard Yang

http://zoo.cs.yale.edu/classes/cs433/

11/13/2018
Admin.

- Programming assignment 4 updated deadlines
  - Part 1: Discussion with instructor or TF checkpoint: Nov. 13; Code checkpoint: 11:55 pm, Nov. 15, 2018
  - Part 2: Design discussion with instructor or TF checkpoint: Nov. 27; Complete code and report due: 1:30 pm, Nov. 29, 2018.
Recap: Transport Reliability Design

- Basic structure of reliability protocol: sliding window protocols, connection management
- Basic analytical technique: execution traces; joint sender/receiver/channel state machine; state invariants
- TCP as an example implementation
  - Hybrid of GBN and SR
  - Full duplex transport
  - Multiple optimizations/adaptation mechanisms
    - Fast retransmit
    - Adaptive RTO
      - mean + variation
    - Adaptive window size (Congestion control)
Recap: Transport Congestion Control Design

- What is congestion control
  - Too high rate can lead to unnecessary long delays, collapse due to waste of resources (e.g., large number of retransmissions, zombie packets)

- Desired properties of congestion control alg
  - distributed algorithm to achieve fairness and efficiency

- Linear control model and requirement

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

$$x_i(t+1) = \begin{cases} 
    a_i + b_i x_i(t) & \text{if } d(t) = \text{no cong.} \\
    a_D + b_D x_i(t) & \text{if } d(t) = \text{cong.}
\end{cases}$$
Recap: Why AIMD Works by Considering State Transition Trace

- Fairness line: \(x_1 = x_2\)
- Efficiency line: \(x_1 + x_2 = C\)
- Underload
- Overload
Intuition: Another Look

- Consider the difference or ratio of the rates of two flows
  - AIAD
  - MIMD
  - MIAD
  - AIMD
Recap: Realizing A(M)IMD: TCP/Reno

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)
Recap: TCP/Reno: Big Picture

TD: Triple duplicate acknowledgements
TO: Timeout
Outline

- Admin and recap
- Transport congestion control/resource allocation
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/reno congestion control
    - design
    - analysis
Objective

- To understand the throughput of TCP/Reno as a function of RTT (RTT), loss rate (p) and packet size
- To better understand system dynamics
- We will analyze TCP/Reno under two different setups
TCP/Reno Throughput Analysis

- 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:

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

- Admin and recap
- Transport congestion control/resource allocation
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/reno congestion control
    - design
    - analysis
      - small fish in a big pond: loss rate given from the environment
TCP/Reno Throughput Modeling

\[
\Delta W = \begin{cases} 
\frac{1}{W} & \text{if the packet is not lost} \\
-\frac{W}{2} & \text{if packet is lost}
\end{cases}
\]

mean of \( \Delta W = (1 - p) \frac{1}{W} + p\left(-\frac{W}{2}\right) = 0 \)

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

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

This is called the TCP throughput sqrt of loss rate law.
Exercise: Application of Analysis

- State of art network link can reach 100 Gbps. Assume packet size 1250 bytes, RTT 100 ms, what is the highest packet loss rate to still reach 100 Gbps?
Outline

- Admin and recap
- Transport congestion control/resource allocation
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/reno congestion control
    - design
    - analysis
      - small fish in a big pond: loss rate given from the environment
        - big fish in small pond: growth causes losses
Total packets sent per cycle = \((W/2 + W)/2 * W/2 = 3W^2/8\)

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

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

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

Question: 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.

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

- If the buffer size at the bottleneck link is very small, what is the link utilization?
Exercise: Small Buffer

- Assume
  - BW: 10 G
  - RTT: 100 ms
  - Packet: 1250 bytes
  - BDP (full window size): 100,000 packets

- A loss can cut window size from 100,000 to 50,000 packets

- To fully grow back
  - Need 50,000 RTTs => 5000 seconds, 1.4 hours
**Discussion**

- Assume a generic AIMD alg: reduce to $\beta W$ after each loss event. What is the avg link utilization when buffer is small?

- If the objective is to maximize link utilization, pick large or small $\beta$?

- Why not pick $\beta$ maximizing utilization?
Outline

- Admin and recap
- Transport congestion control/resource allocation
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
TCP Cubic

- Designed in 2008 by Rhee’ group
- Default for Linux
- Most sockets in MAC appear to use cubic as well
  - `sysctl -a`
    - If you want to see some TCP parameters by a real OS (grep `inet.tcp`)
    - `grep reno cubic`
TCP Cubic Goals

- Improve TCP efficiency over fast, long-distance links with limited buffer

- TCP friendliness: Follows TCP if TCP gives higher rate
Basic Idea I

Minimize waste/effective ratio to increase efficiency

Not too small beta to yield for new competition/fairness
Cubic Window Function

\[ W_{cubic} = C(t - K)^3 + W_{max} \]

\[ K = \frac{3\sqrt{W_{max}\beta}}{C} \]

where \( C \) is a scaling factor, \( t \) is the elapsed time from the last window reduction, and \( \beta \) is a constant multiplication decrease factor.
Basic Idea II: TCP Friendly Rate with Generic $\beta$

Packets per cycle:
$$p = \frac{2\alpha}{(1-\beta)(1+\beta)w^2} \quad w = \sqrt{\frac{2\alpha}{(1-\beta)(1+\beta)p}}$$

$$tput = \frac{W_m S}{RTT} = \frac{S}{RTT} (1+\beta)W = \frac{S}{RTT} \sqrt{\frac{\alpha(1+\beta)}{2(1-\beta)p}}$$

TCP friendly: $\alpha = 3 \frac{1-\beta}{1+\beta}$
Cubic High-Level Structure

- If (received ACK && state == cong avoid)
  - Compute $W_{cubic}(t)$
  - Compute $W_{TCP}(t)$ (form?)
  - Pick the larger one
(a) CUBIC window curves.

(b) Throughput of two CUBIC flows.
Outline

- Admin and recap
- Transport congestion control/resource allocation
  - what is congestion (cost of congestion)
  - basic congestion control alg.
  - TCP/Reno congestion control
  - TCP Cubic
  - TCP/Vegas
TCP/Vegas (Brakmo & Peterson 1994)

- Idea: try to detect congestion by delay before loss
- Objective: not to overflow the buffer; instead, try to maintain a constant number of packets in the bottleneck queue
TCP/Vegas: Key Question

- How to estimate the number of packets queued in the bottleneck queue(s)?
Recall: Little’s Law

- For any system with no or (low) loss.
- Assume
  - mean arrival rate $X$, mean service time $T$, and mean number of requests in the system $W$
- Then relationship between $W$, $X$, and $T$:

$$W = XT$$
Estimating Number of Packets in the Queue
TCP/Vegas CA algorithm

\[ T = T_{\text{prop}} + T_{\text{queueing}} \]

Applying Little's Law:

\[ x_{\text{vegas}} \cdot T = x_{\text{vegas}} \cdot T_{\text{prop}} + x_{\text{vegas}} \cdot T_{\text{queueing}}, \]

where \( x_{\text{vegas}} = \frac{W}{T} \) is the sending rate.

Then number of packets in the queue is

\[ x_{\text{vegas}} \cdot T_{\text{queueing}} = x_{\text{vegas}} \cdot T - x_{\text{vegas}} \cdot T_{\text{prop}} \]

\[ = W - \frac{W}{T} T_{\text{prop}} \quad \text{(value?)} \]
TCP/Vegas CA algorithm

for every RTT
{
    if $W - W/\text{RTT} \leq \frac{\text{RTT}_{\text{min}}}{\alpha}$ then $W$ ++
    if $W - W/\text{RTT} > \frac{\text{RTT}_{\text{min}}}{\alpha}$ then $W$ --
}

for every loss

$W := W/2$
Discussions

- If two flows, one TCP Vegas and one TCP reno run together, how may bandwidth partitioned be among them?

- What are some other key challenges for TCP/Vegas?