Network Transport Layer:
TCP Analysis and BW Allocation Framework

Y. Richard Yang

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

3/30/2016
Admin.

- Programming assignment 3 perf benchmarking
  - If your Zooput is above 100 Mbps, you are invited to attend in person an eval using 40Gbps servers after Monday class

- Programming assignment 4 design review
  - Friday 1:30 - 4:30 pm
  - Sunday 2 - 4:00 pm, 6-8 pm

- Exam 1 (Monday next week)
  - See class page for samples from past
A Summary of Questions

- Basic structure: sliding window protocols
- How to determine the “right” parameters?
  - Timeout
    - mean + variation
  - Sliding window size
    - Goals: distributed adjust window size to achieve fairness and efficiency
    - Linear control parameter: AMI/MD (questions to think about: Do we need more complex control rules?)
TCP/Reno Implementation

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

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

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)
TCP/Reno: Big Picture

TD: Triple duplicate acknowledgements
TO: Timeout
A Session

Question: when cwnd fluctuates widely (i.e., cut to half), why the sending rate does not?
TCP/Reno Queueing Dynamics

- Consider congestion avoidance only

There is a filling and draining of buffer process for each TCP flow. The buffer at the bottleneck keeps pace, as long as it is large enough.
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
    - TCP/reno throughput analysis
Objective

- To understand the throughput of TCP/Reno as a function of RTT (RTT), loss rate (p) and packet size

- We will derive the formula twice, using two setups using two different approaches
TCP/Reno Throughput Modeling

- Given 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

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
      - TCP/reno throughput analysis
        - analysis 1: deterministic
TCP/Reno Throughput Modeling: Relating $W$ with Loss Rate $p$

Consider congestion avoidance only

Assume one packet loss (loss event) per cycle
Total packets sent per cycle $= (W/2 + W)/2 \times W/2 = 3W^2/8$
Thus $p = 1/(3W^2/8) = 8/(3W^2)$

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

$\Rightarrow$ throughput $= \frac{S}{RTT} \times \frac{3}{4} \times \frac{1.6}{\sqrt{p}} = \frac{1.2S}{RTT\sqrt{p}}$
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
      - TCP/reno throughput analysis
        - analysis 1: deterministic
        - analysis 2: random loss
TCP/Reno Throughput Modeling

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

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

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

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

tcp-tput-formula.xlsx
Summary: TCP/Reno Throughput Modeling

- They are all *approximate* modeling
  - the details and the exact numbers are not important
  - the objective is to help us understand TCP better
Problem of TCP/Reno Behavior

\[
\text{cwnd} \quad \text{TD} \quad \text{ssthresh} \quad \text{bottleneck bandwidth} \quad \text{Time}
\]
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
    - TCP/reno congestion control
      - TCP/reno
      - TCP/reno throughput analysis
  - 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 \textit{constant} number of packets in the bottleneck queue
TCP/Vegas: Key Question

- How to estimate the number of packets queued in the bottleneck queue?
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:

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

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

Then number of packets in the queue is

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

\[ = W - \frac{W}{T} T_{\text{prop}} \]
TCP/Vegas CA algorithm

for every RTT

\{
  \text{if } W - W/\text{RTT} \cdot \text{RTT}_{\text{min}} < \alpha \text{ then } W++ \\
  \text{if } W - W/\text{RTT} \cdot \text{RTT}_{\text{min}} > \alpha \text{ then } W--
\}

for every loss

\text{w := w}/2
TCP/Vegas Dynamics

\[ \Delta w_{\text{RTT}} \approx -(w - xRTT_{\text{min}} - \alpha) \]

\[ \Delta w_{\text{unit-time}} = -\left( \frac{w}{RTT} - \frac{x}{RTT} \cdot RTT_{\text{min}} - \frac{\alpha}{RTT} \right) = \frac{x}{RTT} \cdot RTT_{\text{min}} + \frac{\alpha}{RTT} - x \]

\[ \Delta x = \frac{\Delta w_{\text{unit-time}}}{RTT} = \frac{x}{RTT^2} \left( RTT_{\text{min}} + \frac{\alpha}{x} - RTT \right) \]
**TCP/Reno vs. TCP/Vegas**

<table>
<thead>
<tr>
<th></th>
<th>TCP/Reno</th>
<th>TCP/Vegas</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Congestion signal</strong></td>
<td>loss rate $p$</td>
<td>queueuing delay $T_{queueing}$</td>
</tr>
<tr>
<td><strong>Dynamics</strong></td>
<td>$\Delta x = \frac{1}{RTT^2} - p \frac{1}{2} x^2$</td>
<td>$\Delta x = \frac{x}{RTT^2} (RTT_{min} + \frac{\alpha}{x} - RTT)$</td>
</tr>
<tr>
<td><strong>Equilibrium</strong></td>
<td>$x_{reno} = \frac{\alpha_{reno}}{RTT \sqrt{p}}$</td>
<td>$x_{vegas} = \frac{\alpha_{vegas}}{T_{queueing}}$</td>
</tr>
</tbody>
</table>

**Discussion:** Why and why not TCP/Vegas?
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
    - TCP/reno throughput analysis
  - TCP/Vegas

A unifying view of TCP/Reno TCP/Vegas
A Unifying Congestion Measure View

- Both TCP/reno and TCP/Vegas react to congestion measure (loss/delay), which is a signal from the network to the flows reflecting congestion.

- Another way to think of a congestion measure is to think of it as “price”:
  - price goes up as the rate to a link is getting close to capacity
  - the higher the “price”, the lower the rate
Interpreting Congestion Measure

TCP/Reno: \[ \Delta x = \frac{1}{RTT^2} - p \frac{1}{2} x^2 = \frac{1}{2} x^2 \left( \frac{2}{RTT^2 x^2} - p \right) \]

TCP/Vegas: \[ \Delta x = \frac{x}{RTT^2} \left( RTT_{\text{min}} + \frac{\alpha}{x} - RTT \right) = \frac{x}{RTT^2} \left( \frac{\alpha}{x} - T_{\text{queueing}} \right) \]
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
    - TCP/reno throughput analysis
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
    - Network wide resource allocation
So far our discussion is implicitly on a network with a single bottleneck link; this simplifies design and analysis:

- efficiency/optimality (high utilization)
  - fully utilize the bandwidth of the link

- fairness (resource sharing)
  - each flow receives an equal share of the link’s bandwidth
Network Resource Allocation

- It is important to understand and design protocols for a general network topology
  - how will TCP allocate resource in a general topology?
  - how should resource be allocated in a general topology?
Example: TCP/Reno Rates

- Rate:  
  \[ x_1 = 0.26 \]
  \[ x_2 = x_3 = 0.74 \]
Example: TCP/Vegas Rates

Rates: \[ x_1 = \frac{1}{3} \]
\[ x_2 = x_3 = \frac{2}{3} \]
Example: Maximize Throughput

\[
\begin{align*}
\text{max} & \quad \sum_{f} x_f \\
\text{subject to} & \quad x_1 + x_2 \leq 1 \\
& \quad x_1 + x_3 \leq 1
\end{align*}
\]

Optimal: \( x_1 = 0 \)  
\( x_2 = x_3 = 1 \)
Max-min fairness: maximizes the throughput of the flow receiving the minimum (of resources)

- This is a resource allocation scheme used in ATM and some other network resource allocation proposals
Example: Max-Min

\[
\begin{align*}
\max_{x_f \geq 0} & \quad \min \{ x_f \} \\
\text{subject to} & \quad x_1 + x_2 \leq 1 \\
& \quad x_1 + x_3 \leq 1
\end{align*}
\]

Rates: \( x_1 = x_2 = x_3 = 1/2 \)
Network Resource Allocation Using Utility Functions

- A set of flows $F$
- If $x_f$ is the rate of flow $f$, then the utility to flow $f$ is $U_f(x_f)$, where $U_f(x_f)$ is a concave utility function.
- Maximize aggregate utility, subject to capacity constraints

$$\begin{align*}
\text{max} & \quad \sum_{f \in F} U_f(x_f) \\
\text{subject to} & \quad \sum_{f : f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
\text{over} & \quad x \geq 0
\end{align*}$$
Example: Proportional Fairness

\[
\begin{align*}
\max_{x_f \geq 0} & \quad \sum_f \log x_f \\
\text{subject to} & \quad x_1 + x_2 \leq 1 \\
& \quad x_1 + x_3 \leq 1
\end{align*}
\]

\[
U_f \left( x_f \right) = \log \left( x_f \right)
\]

\textbf{Optimal:} \quad x_1 = 1/3 \\
\quad x_2 = x_3 = 2/3
Example 3: a Utility Function

\[
\begin{align*}
\text{max} & \quad -\frac{1}{4x_1} - \frac{1}{x_2} - \frac{1}{x_3} \\
\text{subject to} & \quad x_1 + x_2 \leq 1 \\
& \quad x_1 + x_3 \leq 1
\end{align*}
\]

\[
U_f(x_f) = -\frac{1}{RTT^2 x_f}
\]

Optimal: \(x_1 = 0.26\)
\(x_2 = x_3 = 0.74\)
## Summary: Allocations

<table>
<thead>
<tr>
<th>Objective</th>
<th>Allocation (x₁, x₂, x₃)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/Reno</td>
<td>0.26 0.74 0.74</td>
</tr>
<tr>
<td>TCP/Vegas</td>
<td>1/3 2/3 2/3</td>
</tr>
<tr>
<td>Max Throughput</td>
<td>0 1 1</td>
</tr>
<tr>
<td>Max-min</td>
<td>½ ½ ½</td>
</tr>
<tr>
<td>Max sum log(x)</td>
<td>1/3 2/3 2/3</td>
</tr>
<tr>
<td>Max sum of -1/(RTT² x)</td>
<td>0.26 0.74 0.74</td>
</tr>
</tbody>
</table>

\[
C = 1
\]
Questions

- **Forward engineering:**
  - Derive objective function
  - Derive distributed alg to achieve objective

- **Reverse engineering:** what “objective” functions do TCP/Reno, TCP/Vegas achieve?

<table>
<thead>
<tr>
<th>Objective</th>
<th>Allocation (x1, x2, x3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP/Reno</td>
<td>0.26 0.74 0.74</td>
</tr>
<tr>
<td>TCP/Vegas</td>
<td>1/3 2/3 2/3</td>
</tr>
<tr>
<td>Max throughput</td>
<td>0 1 1</td>
</tr>
<tr>
<td>Max-min</td>
<td>1/2 1/2 1/2</td>
</tr>
<tr>
<td>Max sum log(x)</td>
<td>1/3 2/3 2/3</td>
</tr>
<tr>
<td>Max sum of (-1/(RTT^2 \cdot x))</td>
<td>0.26 0.74 0.74</td>
</tr>
</tbody>
</table>

Mathematical formulation:

\[
\text{max } \sum_{f \in F} U_f(x_f)
\]

subject to

\[
\sum_{f \in F} x_f \leq c_l \text{ for any link } l
\]

over

\[
x \geq 0
\]
Outline

- Recap
- Transport congestion control
  - what is congestion
  - the AIMD alg
  - TCP/reno congestion control
    - TCP/reno
    - TCP/reno throughput analysis
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
  - Network wide resource allocation
    - Framework
      - Axiom derivation of network-wide objective function
Network Bandwidth Allocation Using Nash Bargain Solution (NBS)

- High level picture
  - Given the feasible set of bandwidth allocation, we want to pick an allocation point that is efficient and fair.

- The determination of the allocation point should be based on "first principles" (axioms).
Network Bandwidth Allocation: Feasible Region
Nash Bargain Solution (NBS)

- Assume a finite convex feasible set
- Axioms
Nash Bargain Solution (NBS)

- Assume a finite, convex feasible set
- Axioms
  - Pareto optimality
    - impossibility of increasing the rate of one user without decreasing the rate of another
  - symmetry
    - a symmetric feasible set yields a symmetric outcome
  - invariance of linear transformation
    - the allocation must be invariant to linear transformations of users' rates
  - independence of irrelevant alternatives
    - assume s is an allocation when feasible set is $R$, $s \in T \subseteq R$, then $s$ is also an allocation when the feasible set is $T$
Nash Bargain Solution (NBS)

- Surprising result by John Nash (1951)
  - the rate allocation point is the feasible point which maximizes
    \[ x_1 x_2 \cdots x_F \]
    This is equivalent to maximize
    \[ \sum_f \log(x_f) \]
- In other words, assume each flow \( f \) has utility function \( \log(x_f) \)
- I will give a proof for \( F = 2 \)
  - think about \( F > 2 \)
Nash Bargain Solution

- Assume $s$ is the feasible point which maximizes $x_1 \times x_2$

- Scale the feasible set so that $s$ is at $(1, 1)$
Nash Bargain Solution

Question: after the transformation, is there any feasible point with $x_1 + x_2 > 2$, i.e., is $s$ still the point maximizing $x_1 \times x_2$?
Consider the symmetric rectangle $U$ containing the original feasible set.

According to symmetry and Pareto, $s$ is the allocation when feasible set is $U$.

According to independence of irrelevant alternatives, the allocation of $R$ is $s$ as well.
NBS $\iff$ Proportional Fairness

- Allocation is proportionally fair if for any other allocation, aggregate of proportional changes is non-positive, e.g. if $x_f$ is a proportional-fair allocation, and $y_f$ is any other feasible allocation, then require

$$\sum_{f} \frac{y_f - x_f}{x_f} \leq 0$$
Backup Slides