Network Transport Layer:
Primal-Dual Resource Allocation;
TCP in New Settings

Y. Richard Yang

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

4/6/2016
Outline

- Admin and recap
- Bandwidth allocation framework
- Network overview
Admin.

- Exam 1
  - High - 74
  - Low - 36
  - Solutions and grading rubric to be posted online
    - Please discuss any questions with me and TFs

- Assignment four
  - Questions?

- Projects
Admin: Web Server Benchmarking (2016; Using Zoo Machines)

Performance KB/s (Zoo Machine, 1 Gpbs Link)
Admin: Web Server Benchmarking (Dedicated Machine; 2013)

Apache (2013)

- Apache: 4.925 Gbps
- Fastest student server: 7.530 Gbps
Outline

- Recap
- Transport congestion control
  - What is congestion
  - The AIMD alg
  - TCP/reno congestion control
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
  - Network wide resource allocation
    - Framework
    - Axiom derivation of network-wide objective function
      - Derive distributed algorithm
Outline

- Recap
- Transport congestion control
  - What is congestion
  - The AIMD alg
  - TCP/reno congestion control
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
  - Network wide resource allocation
    - Framework
    - Axiom derivation of network-wide objective function
      - Derive distributed algorithm
## Recap: TCP/Reno and 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>queueing delay $T_{queueing}$</td>
</tr>
<tr>
<td><strong>Dynamics ($x'$)</strong></td>
<td>$\dot{x} = \frac{1}{RTT^2} - \frac{1}{2}px^2$</td>
<td>$\dot{x} = \frac{x}{RTT^2}(RTT_{min} + \frac{\alpha}{x} - RTT)$</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>
Recap: Interpreting Congestion Measure

\[ p_f = \sum_{l \text{ uses } f} q_l \]

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

TCP/Vegas: \[ \dot{x} = \frac{x}{RTT^2} (RTT_{\text{min}} + \frac{\alpha}{x} - RTT) = \frac{x}{RTT^2} \left( \frac{\alpha}{x} - T_{\text{queueing}} \right) \]
Recap: Network-Wide Resource Allocation Examples

<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>$\frac{1}{2}$ $\frac{1}{2}$ $\frac{1}{2}$</td>
</tr>
<tr>
<td>Max sum log(x)</td>
<td>1/3 2/3 2/3</td>
</tr>
<tr>
<td>Max sum of $-1/(RTT^2 , x)$</td>
<td>0.26 0.74 0.74</td>
</tr>
</tbody>
</table>
Roadmap: Resource Allocation Frameworks

- Engineering (design):
  - how to determine objective?
  - given objective, how to design effective alg?

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

- Science/reverse engineering (understand):
  - better understanding of current protocols
Recap: 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 ∈ T ⊆ R, then s is also an allocation when the feasible set is T
Recap: Nash Bargain Solution (NBS)

- Surprising result by John Nash (1951)
  - The rate allocation point is the feasible point which maximizes
    \[ x_1x_2 \cdots x_F \]
  - This is equivalent to maximize
    \[ \sum_f \log(x_f) \]

- In other words, assume each flow \( f \) has utility function \( \log(x_f) \)
Outline

- Recap
- Transport congestion control
  - What is congestion
  - The AIMD alg
  - TCP/reno congestion control
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
- Network wide resource allocation
  - Framework
  - Axiom derivation of network-wide objective function
  - Derive distributed algorithm
Problem

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

Typical setting:
\[
\begin{align*}
\text{max } f(x) \\
\text{s.t. } x \text{ in } S
\end{align*}
\]
Problem

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

Q: Why is it hard to solve the problem using a distributed protocol (i.e., in the Internet)?
A Two-Slide Summary of Constrained Convex Optimization Theory

\[
\begin{align*}
\max & \quad f(x) \\
\text{subject to} & \quad g(x) \leq 0 \\
\text{over} & \quad x \in S
\end{align*}
\]

\(f(x)\) concave
\(g(x)\) linear
\(S\) is a convex set

- Map each \(x\) in \(S\), to \([g(x), f(x)]\)
- For each slope \(q \geq 0\), computes \(f(x) - qg(x)\) of all mapped \([f(x), g(x)]\)

\[
D(q) = \max_{x \in S} \left( f(x) - qg(x) \right)
\]
A Two-Slide Summary of Constrained Convex Optimization Theory

\[
\begin{align*}
\max & \quad f(x) \\
\text{subject to} & \quad g(x) \leq 0 \\
\text{over} & \quad x \in S
\end{align*}
\]

- \( f(x) \) concave
- \( g(x) \) linear
- \( S \) is a convex set

\[
D(q) = \max_{x \in S} \left( f(x) - qg(x) \right)
\]

- \( D(q) \) is called the dual;
- \( q \ (\geq 0) \) are called prices in economics
- \( D(q) \) provides an upper bound on obj.
- According to optimization theory:
  - When \( D(q) \) achieves minimum over all \( q \ (\geq 0) \), then the optimization objective is achieved.
Dual of the Primal

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

\[D(q) = \max_{x_f \geq 0} \left( \sum_f U_f (x_f) - \sum_l q_l \left( \sum_{f: \text{uses } l} x_f - c_l \right) \right)\]
Decomposition

Assume each link \( l \) has non-negative congestion signal \( q_l \), consider the dual \( D(q) \)

\[
D(q) = \max_{x_f \geq 0} \left( \sum_f U_f(x_f) - \sum_l q_l \left( \sum_{f: \text{uses } l} x_f - c_l \right) \right)
\]

\[
= \max_{x_f \geq 0} \sum_f \left( U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_l q_l c_l
\]

\[
= \sum_f \max_{x_f \geq 0} \left( U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_l q_l c_l
\]
Distributed Optimization: User Problem

Given network-wide signal per unit rate $p_f$ (=sum of $q_i$ along the path), flow $f$ chooses rate $x_f$ to maximize:

$$\max_{x_f} \ U_f(x_f) - x_f p_f$$

over $x_f \geq 0$

Using the network signals, the optimization problem of each user is **decoupled**: independent of each other!
Distributed Optimization: User Problem

How should flow $f$ adjust $x_f$ locally?

$$\Delta x_f \propto U'_f (x_f) - p_f$$

At equilibrium (i.e., at optimal), $x_f$ satisfies:

$$U'_f (x_f) - p_f = 0$$
Interpreting Congestion Measure

\[ p_f = \sum_{f \text{ uses } l} q_l \]

\[ \Delta x_f \propto U'_f (x_f) - p_f \]
Distributed Optimization: Network Problem

\[ D(q) = \sum_{f} \max_{x_f \geq 0} \left( U_f(x_f) - x_f \sum_{l: f \text{ uses } l} q_l \right) + \sum_{l} q_l c_l \]

Assume after all flows have picked their optimal rates given congestion signal, the network (i.e., link \( l \)) adjusts the link signals \( q_l \):

\[
\min_{q \geq 0} \sum_{l} q_l \left( c_l - \sum_{f: f \text{ uses } l} x_f \right)
\]
Distributed Optimization: Network Problem

\[
\min_{q \geq 0} D(q) = \sum_l q_l (c_l - \sum_{f : f \text{ uses } l} x_f )
\]

how should link \( l \) adjust \( q_l \) locally?

\[
\Delta q_l \propto -\frac{\partial D(q)}{q_l}
\]

\[
\frac{\partial}{\partial q_l} D(q) = c_l - \sum_{f : \text{uses } l} x_f
\]

\[
\Delta q_l \propto \sum_{f : \text{uses } l} x_f - c_l
\]
Decomposition

- **SYSTEM(U):**

  \[
  \text{max} \sum_{f \in F} U_f(x_f) \\
  \text{subject to} \sum_{f : f \text{ uses } l} x_f \leq c_l \text{ for any link } l \\
  \text{over } x \geq 0
  \]

- **USER}_f:*

  \[
  \text{max} U_f(x_f) - x_f p_f \quad \text{over } x_f \geq 0
  \]

- **NETWORK:**

  \[
  \min_{q \geq 0} \tilde{D}(q) = \sum_l q_l (c_l - \sum_{f : f \text{ uses } l} x_f)
  \]
Outline

- Recap
- Transport congestion control
  - What is congestion
  - The AIMD alg
  - TCP/reno congestion control
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
  - Network wide resource allocation
    - Framework
    - Axiom derivation of network-wide objective function
    - Derive distributed algorithm
      - TCP/Reno, TCP/Vegas revisited
TCP/Reno Dynamics

\[ \Delta x_f \propto U'_f (x_f) - p_f \]

\[ \dot{x} = \frac{x^2}{2} \left( \frac{2}{RTT^2 x^2} - p \right) \]

\[ U'_f (x_f) - p_f \]

\[ \Rightarrow U'_f (x_f) = \left( \frac{\sqrt{2}}{x_f RTT} \right)^2 \]

\[ \Rightarrow U_f (x_f) = - \frac{2}{RTT^2 x_f} \]
TCP/Vegas Dynamics

\[ \dot{x} = \frac{x}{RTT^2} \left( \frac{\alpha}{x} - (RTT - RTT_{\text{min}}) \right) \]

\[ U'_f (x_f) - p_f \]

\[ \Rightarrow U'_f (x_f) = \frac{\alpha}{x} \quad \Rightarrow U_f (x_f) = \alpha \log(x_f) \]
Summary: TCP/Vegas and TCP/Reno

- Pricing signal is queueing delay $T_{queueing}$

\[
X_f = \frac{\alpha}{T_{queueing}}
\]

$U'_f(x_f) = T_{queueing}$

\[ \Rightarrow U'_f(x_f) = \frac{\alpha}{x_f} \]

\[ \Rightarrow U_f(x_f) = \alpha \log(x_f) \]

- Pricing signal is loss rate $p$

\[
X_f = \frac{\alpha}{RTT \sqrt{p}}
\]

$U'_f(x_f) = p$

\[ \Rightarrow U'_f(x_f) = \left( \frac{\alpha}{x_f RTT} \right)^2 \]

\[ \Rightarrow U_f(x_f) = -\frac{\alpha'}{RTT^2 x_f} \]
Summary: Resource Allocation Framework

- Engineering (design):
  - how to determine objective?
  - given objective, how to design effective alg?

- Science/reverse engineering (understand):
  - better understanding of current protocols

- Additional pointers
  - http://www.statslab.cam.ac.uk/~frank/pf/
Outline

- Recap
- Transport congestion control
  - What is congestion
  - The AIMD alg
  - TCP/reno congestion control
  - TCP/Vegas
  - A unifying view of TCP/Reno TCP/Vegas
  - Network wide resource allocation
    - Framework
      - Axiom derivation of network-wide objective function
      - Derive distributed algorithm
  - New protocols
Motivation

TCP/Reno growth function

Grows linearly throughout.
Motivation: Is there a faster way?
TCP BIC Algorithm

- **Setting**
  - $W_{\text{max}} = \text{cwnd size before reduction}$
    - Too big
  - $W_{\text{min}} = \beta W_{\text{max}}$ - just after reduction, where $\beta$ is multiplicative decrease factor
    - Small

- **Basic idea**
  - binary search between $W_{\text{max}}$ and $W_{\text{min}}$
TCP BIC Algorithm: Issues

- Pure binary search (jump from $W_{\text{min}}$ to $(W_{\text{max}} + W_{\text{min}})/2$) may be too aggressive
  - Use a large step size $S_{\text{max}}$

- What if you grow above $W_{\text{max}}$?
  - Use binary growth (slow start) to probe more
TCP BIC Algorithm

Packet loss event

- **Additive Increase**
  - $W_{\text{min}} = \beta W_{\text{max}}$
  - $W_{\text{min}} + S_{\text{max}}$
  - $W_{\text{min}} + S_{\text{min}}$

- **Binary Search**
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2

- **Slow Start**
  - $W_{\text{max}} + 2S_{\text{max}}$
  - $W_{\text{max}} + 3S_{\text{max}}$

- **Additive Inc.**
  - Max Probing
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2

- **Event**: Packet loss

- **Equations**:
  - $W_{\text{min}} = \beta W_{\text{max}}$
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2
  - midpoint = ($W_{\text{min}} + W_{\text{max}}$)/2

- **Ranges**:
  - $W_{\text{min}}$
  - $W_{\text{min}} + S_{\text{min}}$
  - $W_{\text{min}} + S_{\text{max}}$
  - $W_{\text{max}}$

- **Criteria**:
  - midpoint - $W_{\text{min}}$ < $S_{\text{min}}$
  - midpoint - $W_{\text{min}}$ > $S_{\text{max}}$
TCP BIC Algorithm

while (cwnd < Wmax) {
    if ( (midpoint - Wmin) > Smax )
        cwnd = cwnd + Smax
    else
        if ( (midpoint - Wmin) < Smin)
            cwnd = Wmax
        else
            cwnd = midpoint
    if (no packet loss)
        Wmin = cwnd
    else
        Wmin = β*cwnd
    Wmax = cwnd
    midpoint = (Wmax + Wmin)/2
}
TCP BIC Algorithm: Probe

```plaintext
while (cwnd >= W_{max}){
    if (cwnd < W_{max} + S_{max})
        cwnd = cwnd + S_{min}
    else
        cwnd = cwnd + S_{max}
    if (packet loss)
        W_{min} = \beta \cdot cwnd
        W_{max} = cwnd
}
```
TCP BIC - Summary

Packet loss event

Additive Increase

Binary Search

W_{max}

W_{max}

+ S_{max}

Jump to midpoint

Additive Increase

Binary Increase

Slow Start

Additive Increase

Max Probing

Time

+ S_{min}

+ S_{max}
TCP BIC in Action
TCP BIC Analysis

- Advantages
  - Faster convergence at large gap
  - Slower growth at convergence to avoid timeout

- Issues
  - Still depend on RTT
  - Complex growth function

TCP Cubic

- cwnd = $C(t - K)^3 + W_{max}$, where
  - $W_{max}$ = cwnd before last reduction
  - $C$ scaling factor
  - $t$ is the time elapsed since last window reduction
  - $K = \sqrt[3]{W \beta / C}$
  - $\beta$ multiplicative decrease factor
TCP CUBIC

Packet loss event

Fast growth upon reduction

Around $W_{\text{max}}$, window growth almost becomes zero

Cubic starts probing for more bandwidth

Max Probing

Steady State Behavior

$W_{\text{max}}$
TCP CUBIC Advantages

- **Good RTT fairness**
  - Growth dominated by $t$, competing flows have same $t$ after synchronized packet loss

- **Real-time dependent**
  - Similar to BIC but linear increases are time dependent
  - Does not depend on ACK's like TCP/ Reno

- **Scalability**
  - Cubic increases window to $W_{\text{max}}$ (or its vicinity) quickly and keeps it there longer
TCP CUBIC Drawbacks

- **Slow Convergence**
  - Flows with higher cwnd are more aggressive initially
  - => Prolonged unfairness between flows

- More details:
Summary

- Many aspects of TCP can be studied, for example TCP under wireless (LTE)
Outline

- Admin and recap
- Network resource allocation framework
  - Network overview
Network Layer

- Transport packet from source to dest.
- Network layer in every host, router

Basic function:
- determine route taken by packets of a flow, and move the packets along the route
Network Layer: Complexity Factors

- **For network providers**
  - efficiency of routes
  - policy control on routes
  - scalability

- **For users: quality of services**
  - guaranteed bandwidth?
  - preservation of inter-packet timing (no jitter)?
  - loss-free delivery?
  - in-order delivery?

- **Interaction between users and network providers**
  - signaling: congestion feedback/resource reservation
## Network Layer Quality of Service

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Service Model</th>
<th>Guarantees?</th>
<th>Congestion feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bandwidth</td>
<td>Loss</td>
</tr>
<tr>
<td>Internet</td>
<td>best effort</td>
<td>none</td>
<td>no</td>
</tr>
<tr>
<td>ATM</td>
<td>CBR</td>
<td>constant rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>VBR</td>
<td>guaranteed rate</td>
<td>yes</td>
</tr>
<tr>
<td>ATM</td>
<td>ABR</td>
<td>guaranteed minimum</td>
<td>no</td>
</tr>
<tr>
<td>ATM</td>
<td>UBR</td>
<td>none</td>
<td>no</td>
</tr>
</tbody>
</table>

- Internet model being extended: Intserv, Diffserv
- multimedia networking

ATM: Asynchronous Transfer Mode; CBR: Constant Bit Rate; V: Variable; A: available; U: User
Current Internet Network Layer

Network layer functions:

- **Routing protocols**
  - path selection
  - e.g., RIP, OSPF, BGP

- **Control protocols**
  - error reporting
  - e.g. ICMP

- **Control protocols**
  - router “signaling”
  - e.g. RSVP

- **Network layer protocol** (e.g., IP)
  - addressing conventions
  - packet format
  - packet handling conventions
Routing: Overview

Routing

_goal: determine “good” paths (sequences of routers) thru networks from source to dest._

Graph abstraction for the routing problem:

- graph nodes are routers
- graph edges are physical links
  - links have properties: delay, capacity, $ cost, policy
Routing Design Space

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
  - how many paths from source s to destination d?
    - multi-path routing
    - single path routing
  - will routing adapt to network traffic demand?
    - adaptive routing
    - static routing
  - ...

- Robustness
- Optimality
- Simplicity
Routing Design Space: User-based, Multipath, Adaptive

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
      - network routing: networks make decision
  - how many paths from source s to destination d?
    - multi-path routing
      - single path routing
  - will routing adapt to network traffic demand?
    - adaptive routing
      - static routing
  - ...
User Optimal, Multipath, Adaptive

- User optimal: users pick the shortest routes (selfish routing)

```
Flow = .5
```

this flow is envious!

```
Flow = 1
```

Braess’s paradox
Price of Anarchy

For a network with linear latency functions

\[ \text{total latency of user (selfish) routing for given traffic demand} \leq \frac{4}{3} \]

\[ \text{total latency of network optimal routing for the traffic demand} \]
Price of Anarchy

For any network with continuous, non-decreasing latency functions →

total latency of user (selfish) routing for given traffic demand ≤

total latency of network optimal routing for twice traffic demand
Routing Design Space: Internet

- Routing has a large design space
  - who decides routing?
    - source routing: end hosts make decision
    - network routing: networks make decision
      - (applications such as overlay and p2p are trying to bypass it)
  - how many paths from source s to destination d?
    - multi-path routing
      - single path routing (with small amount of multipath)
  - will routing adapt to network traffic demand?
    - adaptive routing
      - static routing (mostly static; adjust in larger timescale)
  - ...

- Robustness
- Optimality
- Simplicity
Backup Slides
Another Decomposition

- **SYSTEM(U):**
  \[
  \begin{align*}
  & \text{max } \sum_{f \in F} U_f \left( x_f \right) \\
  & \text{subject to } \sum_{f \mid f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
  & \text{over } x \geq 0
  \end{align*}
  \]

- **USER}_f(U_f; p_f)**
  \[
  \begin{align*}
  & \text{max } U_f \left( \frac{w_f}{p_f} \right) - w_f \\
  & \text{over } w_f \geq 0
  \end{align*}
  \]

- **NETWORK}(w_f)**
  \[
  \begin{align*}
  & \text{max } \sum_{f \in F} w_f \log_f \left( x_f \right) \\
  & \text{subject to } \sum_{f \mid f \text{ uses link } l} x_f \leq c_l \text{ for any link } l \\
  & \text{over } x \geq 0
  \end{align*}
  \]
Decomposition Theorem

- There exist vectors $p$, $w$ and $x$ such that
  1. $w_f = p_f x_f$ for $f \in F$
  2. $w_f$ solves $\text{USER}_f(U_f; p_f)$
  3. $x$ solves $\text{NETWORK}(w)$

- The vector $x$ then also solves $\text{SYSTEM}(U)$. 