A Game-Theoretic Analysis of TCP

Aditya Akella, Richard Karp, Christos Papadimitriou, Srinivasan Seshan, Scott Shenker

CMU/ICSI/UC Berkeley
1980s: Network Collapse

In the 80s’s, naïve behavior caused the network to collapse.
Socially responsible congestion control implemented at end-points was given credit for saving the Internet
Salvation?

Can the network survive with greedy (but intelligent) behavior?
Why Bother?

- If greed is bad, today’s Internet is stable because –
  - End-points are consciously social and/or
  - It is hard to modify end-hosts to be greedy

We may need mechanisms to monitor, dissuade aggressive behavior

- If not, we need no such mechanism
  - Can rely on end-point behavior for efficient operation
Outline

- The TCP Game
- Results for the TCP Game
- Mechanisms for Nash Equilibrium
The TCP Game

- **TCP Game**
  - Flows attempt to maximize application throughput
    - Flows modify their AIMD parameters \((\alpha, \beta)\)
    - Must still provide reliability
  
- **What happens at Nash Equilibrium?**
  - No flow can gain in throughput by unilaterally changing its parameter choice
The TCP Game

- Analyze a simplified version of this Game for...
  - Parameters at the Nash Equilibrium
  - Efficiency at the Nash Equilibrium
    - Link Goodput and per-flow Loss rate
- Study symmetric Nash Equilibria only
Factors Affecting the Nash Equilibrium

(1) End-point’s loss recovery mechanism
   - Reno vs. SACK (primitive vs. modern)
   - Depends on TCP implementation

(2) Loss assignment at routers
   - Bursty loss assignment vs. randomized uniform losses

(3) Congestion control parameters of the flows
   - How flows are allowed to vary their parameters
   - Under complete control of end-point

End-points show greed by adjusting factor (3) alone. Factors (1), (2) are part of the environment.
Outline

- The TCP Game
- Results for the TCP Game
- Mechanisms for Nash Equilibrium
Results – Road-map

- First, consider FIFO droptail buffers
  - Most wide-spread in today’s Internet
  - Efficiency at Nash Equilibrium for Tahoe, Reno, SACK-style loss recovery

- Then, discuss RED buffers briefly
  - As above

- Put the results together
FIFO Droptail Buffering

- Droptail buffers punish bursty behavior
  - Unintentionally so
  - Observed by designers of RED AQM
  - Flows with bursty transmission incur losses proportional to their burstiness
    - AIMD flows incur losses in bursts of size $\sim \alpha$ (AI parameter)
Results for FIFO Droptail Buffers – A Sample

- Greedy flows don’t gain by using large $\alpha$’s
  - Flows observe burst losses
  - Reno’s severe reaction (time-outs) kicks in

\[ \alpha_1 \ldots \alpha_{n-1} = 1 \]

Reno-style loss recovery (flows vary $\alpha$, keeping $\beta=0.5$)

<table>
<thead>
<tr>
<th>$\alpha_E$</th>
<th>Link Goodput</th>
<th>Per-flow Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100%</td>
<td>0.15%</td>
</tr>
</tbody>
</table>

Aditya Akella (aditya@cs.cmu.edu)
Results for FIFO Droptail Buffers –
A Sample

- Greedy flows gain by using $\beta \rightarrow 1$
  - No burst losses since $\alpha = 1$

Reno-style loss recovery
(flows vary $\beta$, keeping $\alpha = 1$)

<table>
<thead>
<tr>
<th>$\beta_E$</th>
<th>Link Goodput</th>
<th>Per-flow Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.98</td>
<td>100%</td>
<td>1.5%</td>
</tr>
</tbody>
</table>

Aditya Akella (aditya@cs.cmu.edu)
Results for FIFO Droptail Buffers – A Sample

Reno-style loss recovery
(flows vary $\alpha$, $\beta$ together)

<table>
<thead>
<tr>
<th>$(\alpha_E, \beta_E)$</th>
<th>Link Goodput</th>
<th>Per-flow Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 0.98)</td>
<td>100%</td>
<td>1.5%</td>
</tr>
</tbody>
</table>

- Nash Equilibrium is efficient!
  - Goodput is high and loss rate is low
  - Greedy behavior might work out
- But unfair
  - Since $\beta \to 1$ (AIAD)
RED Buffering

- RED buffers “spread” losses uniformly across flows
  - Identical loss %-age across flows irrespective of parameters used
  - Greater greed of a few flows causes a small increase in overall loss rate
  - Bursty flows do not experience burst losses, unlike droptail buffers
Results for RED Buffers – A Sample

- Aggression is always good
  - TCP SACK $\rightarrow$ high loss rate doesn’t affect goodput
  - RED $\rightarrow$ greater aggression will cause minor increase in overall loss rate

<table>
<thead>
<tr>
<th>$\alpha_E$</th>
<th>Link Goodput</th>
<th>Per-flow Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>96%</td>
<td>5%</td>
</tr>
</tbody>
</table>
SACK-style loss recovery
Flows vary $\alpha, \beta$ together

<table>
<thead>
<tr>
<th>$(\alpha_E, \beta_E)$</th>
<th>Link Goodput</th>
<th>Per-flow Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>(23, 0.98)</td>
<td>96%</td>
<td>5.70%</td>
</tr>
</tbody>
</table>

- Nash Equilibrium is inefficient
  - Parameter setting is very aggressive
    - Loss rate is high
  - Potential congestion collapse!
## Results for the TCP Game – A Summary

<table>
<thead>
<tr>
<th></th>
<th>Droptail</th>
<th>RED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tahoe</td>
<td>$\alpha_E \rightarrow \text{high, } \beta_E \rightarrow 1$, Inefficient</td>
<td>$\alpha_E \rightarrow \text{high, } \beta_E \rightarrow 1$, Inefficient</td>
</tr>
<tr>
<td>Reno</td>
<td>$\alpha_E = 1, \beta_E \rightarrow 1$, Efficient, Unfair</td>
<td>$\alpha_E \rightarrow \text{high, } \beta_E \rightarrow 1$, Inefficient</td>
</tr>
<tr>
<td>SACK</td>
<td>$\alpha_E \rightarrow \text{high, } \beta_E \rightarrow 1$, Inefficient</td>
<td>$\alpha_E \rightarrow \text{high, } \beta_E \rightarrow 1$, Inefficient</td>
</tr>
</tbody>
</table>
Discussion

Question: Does selfish congestion control endanger network efficiency?

Common Intuition: Yes, since flows would always gain from being more aggressive.

Our Answer: Not necessarily true!

- In the traditional setting (Reno end-points and droptail routers), network operates fine despite selfish behavior
- Selfish behavior very detrimental with modern loss recovery and queue management schemes
Outline

- The TCP Game
- Results for the TCP Game
- Mechanisms for Nash Equilibrium
Mechanisms for Nash Equilibrium

- We need mechanisms to explicitly counter aggressive behavior
- Has been a hot topic in the past
  - Fair Queuing discourages aggressive behavior
    - But needs per-flow state
  - RED-PD, AFD etc. explored lighter mechanisms
    - Aim to ensure fair bandwidth allocation

Our requirement is less stringent:

How much preferential dropping is needed to ensure a reasonable Nash Equilibrium?
CHOKe+ -- A Simple, Stateless Scheme

- A small modification to RED is enough

**CHOKe+**
- Simple, stateless
- Provides just the right amount of punishment to aggressive flows
- Makes marginal advantage from greed insignificant
- E.g. SACK flows varying $\alpha$

<table>
<thead>
<tr>
<th>$\alpha_E$</th>
<th>Goodput</th>
<th>Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>97%</td>
<td>2.74%</td>
</tr>
</tbody>
</table>

Aditya Akella (aditya@cs.cmu.edu)
CHOKe+ (Cont.)

- \( \beta \to 1 \) at Nash Equilibrium in all cases
  - \( \beta < 1 \) impossible to ensure without Fair Queuing
- But, CHOKe+ encourages \( \beta < 1 \)
  - Makes aggressive \( \beta \) a risky choice
  - With SACK flows \( \beta = 0.74 \) at Nash Equilibrium

<table>
<thead>
<tr>
<th>( \beta_E )</th>
<th>Goodput</th>
<th>Loss rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.74</td>
<td>100%</td>
<td>2.42%</td>
</tr>
</tbody>
</table>

Aditya Akella (aditya@cs.cmu.edu)
Summary

- Greedy congestion control may not always lead to inefficient operation
  - Traditional Reno host-droptail router setting
- Unfortunately, greedy behavior is bad in most other situations
- Fortunately, it is possible to ensure a desirable Nash Equilibrium via simple, stateless mechanisms
Back-up

- Back-up
  - Back-up
  - Back-up
CHOKe+

- CHOKe would have worked
  - But, enforces too high a drop rate
  - Underutilization at low levels of multiplexing
  - CHOKe+ fixes this problem
The CHOKe+ Algorithm

- For each incoming packet P
  - Pick $k$ packets at random from queue
  - Let $m$ be # packets from the same flow as P
  - Let $0 \leq \gamma_2 < \gamma_1 \leq 1$ be constants
  - If $m > \gamma_1 k$, P and the $m$ packets are dropped
  - Else if $\gamma_2 k \leq m < \gamma_1 k$, drop P and the $m$ packets only if RED were to drop P
  - Else just drop P according to RED
Why AIMD?

- Analysis is more generic than meets the eye
  - Conclusions hold for other congestion control schemes
  - Burstiness is a property of probing
- Widely employed
Why not Change Loss Recovery?

- Historical evaluation
- Very difficult to change
  - Sometimes need bilateral (protocol) support
  - Needs many implementation changes
  - Many design decisions were influenced by system requirements