Analytical Model for Congestion Control and Throughput with TCP CUBIC connections

Sudheer Poojary, Vinod Sharma

IISc, Bangalore

6 Dec, 2011
1 Introduction

2 TCP CUBIC: Overview

3 TCP CUBIC: Constant RTT

4 TCP CUBIC: Variable RTT

5 Multiple TCP Connections

6 Conclusions and Future Work
Need for speed

- Traditionally TCP (TCP Reno, TCP NewReno, TCP SACK) uses an AIMD algorithm for congestion control.
- In the AI phase the window grows linearly.
- On detection of loss, there is a multiplicative decrease.
- For high speed networks AIMD too slow; inefficient link usage.
The high speed TCP algorithms

- FAST TCP
  - Queuing delay indicates congestion.
  - Scalable window growth for fast window increase.
- HTCP
  - In AI phase additive increase rate based on time since last packet drop.
- TCP Compound
  - Has both delay based and loss based components.
- TCP CUBIC
TCP CUBIC: Features

- Like HTCP, uses time elapsed since last packet drop to update its window.
- Is default Linux TCP algorithm since 2006.
- The window growth function is CUBIC.
  - Aggressive when away from the previous congestion window.
  - Slows down when close to previous congestion window.
TCP CUBIC: Algorithm description

Window growth function

\[ W_{cubic}(W_0, t) = C(t - K)^3 + W_0. \]  \hspace{1cm} (1)

- \( W_0 \): window size at the last congestion epoch
- \( t \) is the time since last congestion
- \( K = \sqrt[3]{\frac{W_0 \beta}{C}} \)
- \( \beta \): the multiplicative drop factor
- If loss, the window size is reduced by a factor of \( (1 - \beta) \).

Also uses

\[ W_{reno}(W_0, t) = W_0(1 - \beta) + 3 \frac{\beta}{2 - \beta} \frac{t}{RTT}. \]  \hspace{1cm} (2)

if \( W_{reno} > W_{cubic} \), then it uses (2) else it uses (1).
Analytical Model: Assumptions

- Random Bernoulli packet losses.
  - Reasonable when end systems use wireless links.
- One router
- One TCP connection with enough link capacity: no queuing, constant RTT
- All ACKs of a window reach TCP source at the end of RTT.
- The TCP sending rate is constrained by buffer at the receiver.
### Analytical Model: Notation

- $W_k$: Window size at the end of $k^{th}$ RTT.
- $W'_k$: Window size at the congestion epoch preceding the $k^{th}$ RTT.
- $D_k$: Time elapsed since last congestion (in multiples of RTT).
- $W_{max}$: Maximum congestion window size.

<table>
<thead>
<tr>
<th>$D_k$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_k$</td>
<td>$W_0$</td>
<td>$W_1$</td>
<td>$W_2$</td>
<td>$W_3$</td>
<td>$W_4$</td>
<td>$W_5$</td>
<td>$W_6$</td>
<td>$W_7$</td>
</tr>
<tr>
<td>$W'_k$</td>
<td>$W_0$</td>
<td>$W_0$</td>
<td>$W_0$</td>
<td>$W_2(1-\beta)$</td>
<td>$W_2(1-\beta)$</td>
<td>$W_2(1-\beta)$</td>
<td>$W_2(1-\beta)$</td>
<td>$W_6(1-\beta)$</td>
</tr>
</tbody>
</table>

* : Packet Drop
Analytical Model Description

The window size at the end of $k^{th}$ RTT

$$W_k = \min(W_{\text{max}}, \max(W_{\text{cubic}}(w', d), W_{\text{reno}}(w', d)))$$  \hspace{1cm} (3)

where

$$W'_k = w', \ D_k = d$$

The process evolves as follows

- If no loss in the $k^{th}$ RTT,

  $$W'_{k+1} = W'_k, \ D_{k+1} = D_k + 1.$$  \hspace{1cm} (4)

- If loss

  $$W'_{k+1} = W_k(1 - \beta), \ D_{k+1} = 0.$$  \hspace{1cm} (5)
Analytical Model

- The probability of at least one drop in $k^{th}$ RTT is
  $$1 - (1 - p)^{W_k}.$$ 
- $\{(W'_k, D_k)\}$ forms a discrete time countable state Markov chain.
- The MC is aperiodic, positive recurrent with unique stationary distribution $\pi$.
- We have

\[
E[W] = \sum_{w' = 1}^{(1-\beta)W_{\text{max}}} \left( \sum_{d = 0}^{d = \infty} W_k(w', d)\pi(w', d) \right)
\]

\(\triangleq g_p(RTT).\)
Results: TCP CUBIC Constant RTT

Figure: Average window sizes obtained using the Analytical model
TCP CUBIC: non-negligible queuing delay

- Non-negligible queuing delay.
- $E[S]$: the average sojourn time in queue.
- $E[RTT] = \Delta + E[S]$.
- Use M/G/1 approximation for $E[S]$.
- $E[W]$ obtained from the constant RTT model using $E[RTT]$:

$$E[W] = g_p(E[RTT])$$  \hspace{0.5cm} (7)
TCP CUBIC throughput with non-negligible queuing delay

- \( s \): packet lengths
- \( \lambda \): throughput in packets/sec
- \( C \): link capacity in bps.

Using the M/G/1 results and Little’s law.

\[
E[S] = \frac{\lambda E[s^2]}{2C^2(1 - \rho)} + \frac{E[s]}{C} \tag{8}
\]

- \( \rho = \frac{\lambda E[s]}{C} \)

\[
\lambda = \frac{E[W]}{E[RTT]} \tag{9}
\]

\[
E[RTT] = \Delta + E[S] \tag{10}
\]

Solve (8), (9), (10) simultaneously for \( E[S] \), \( \lambda \), \( E[RTT] \)
Results: TCP CUBIC Variable RTT, Average window size

\[ E[W] \text{ vs } \Delta \text{ for TCP CUBIC, link capacity = 10Mbps, link packet error rate } = p \]

- ○ NS2, \( p = 0.01 \)
- ▲ MGI model, \( p = 0.01 \)
- ★ NS2, \( p = 0.005 \)
- □ MGI model, \( p = 0.005 \)
Results: TCP CUBIC Variable RTT, Throughput

Throughput vs $\Delta$ for TCP CUBIC, link capacity = 10Mbps, link packet error rate = p

- NS2, $p = 0.01$
- MG1 model, $p = 0.01$
- NS2, $p = 0.005$
- MG1 model, $p = 0.005$
Multiple TCP Connections

- Multiple TCP connections share a link.
- PER on Connection $i$ is $= p_i$.
- Propagation delay of connection $i = \Delta_i$.

**Figure:** System with $N$ TCP connections sharing a link.
Throughput for multiple TCP connections

\[ E[RTT_i] = \frac{\lambda E[s^2]}{2C^2(1 - \rho)} + \frac{E[s]}{C} + \Delta_i. \]  

where \( E[s] \) and \( E[s^2] \) are the overall mean packet length and its second moment.

\( \lambda \) = the overall throughput in packets/sec.

\( \lambda_i \) = throughput of TCP connection \( i \).

\[ E[s] = \sum_i \frac{\lambda_i}{\lambda} E[s_i], \quad E[s^2] = \sum_i \frac{\lambda_i}{\lambda} E[s_i^2], \quad \rho = \frac{\lambda E[s]}{C}. \]
Throughput for multiple TCP connections

\[ E[W_i] = g_{p_i}(E[RTT_i]) \]  \hspace{1cm} (13)

\[ \lambda_i = \frac{E[W_i]}{E[RTT_i]} \cdot \]  \hspace{1cm} (14)

- We solve (11), (13) and (14) simultaneously to obtain 
  \( E[RTT_i] \), \( E[W_i] \) and \( \lambda_i \) for \( i = 1, 2, \ldots, N \).
- Can also include TCP Reno connections.
  - Then \( g_{p_i} \) in (13) is not needed: \( E[W_i] \) is function of \( p_i \) but independent of RTT.
### Results: Multiple Connections, Avg. Window Size

**Table:** Average window sizes for three TCP CUBIC connections with bottleneck link capacity = 5 Mbps

<table>
<thead>
<tr>
<th>Connection $i$</th>
<th>$p_i$</th>
<th>$\Delta_i$ (sec)</th>
<th>$EW_i$ (NS2)</th>
<th>$EW_i$ (Theory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.005</td>
<td>0.5</td>
<td>49.6</td>
<td>45.8</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>20.1</td>
<td>19.2</td>
</tr>
<tr>
<td>3</td>
<td>0.005</td>
<td>0.04</td>
<td>18.9</td>
<td>17.8</td>
</tr>
<tr>
<td>1</td>
<td>0.005</td>
<td>0.1</td>
<td>19.4</td>
<td>18.6</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>19.4</td>
<td>18.6</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.1</td>
<td>13.3</td>
<td>12.7</td>
</tr>
<tr>
<td>1</td>
<td>0.003</td>
<td>0.1</td>
<td>26.2</td>
<td>25.1</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>19.6</td>
<td>18.7</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.1</td>
<td>13.5</td>
<td>12.8</td>
</tr>
<tr>
<td>1</td>
<td>0.003</td>
<td>0.04</td>
<td>25.0</td>
<td>23.3</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>20.6</td>
<td>19.5</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.5</td>
<td>29.1</td>
<td>27.3</td>
</tr>
</tbody>
</table>
Results: Multiple Connections, Throughput

Table: Throughput for three TCP CUBIC connections with bottleneck link capacity = 5 Mbps

<table>
<thead>
<tr>
<th>Connection i</th>
<th>$p_i$</th>
<th>$\Delta_i$ (sec)</th>
<th>$\lambda_i$ (NS2)</th>
<th>$\lambda_i$ (Theory)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.005</td>
<td>0.5</td>
<td>94.5</td>
<td>88.5</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>168.6</td>
<td>163.9</td>
</tr>
<tr>
<td>3</td>
<td>0.005</td>
<td>0.04</td>
<td>323.0</td>
<td>311.9</td>
</tr>
<tr>
<td>1</td>
<td>0.005</td>
<td>0.1</td>
<td>186.2</td>
<td>176.8</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>186.9</td>
<td>176.8</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.1</td>
<td>126.7</td>
<td>121.0</td>
</tr>
<tr>
<td>1</td>
<td>0.003</td>
<td>0.1</td>
<td>241.5</td>
<td>232.9</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>181.3</td>
<td>173.4</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.1</td>
<td>124.0</td>
<td>118.3</td>
</tr>
<tr>
<td>1</td>
<td>0.003</td>
<td>0.04</td>
<td>376.8</td>
<td>364.0</td>
</tr>
<tr>
<td>2</td>
<td>0.005</td>
<td>0.1</td>
<td>161.8</td>
<td>157.6</td>
</tr>
<tr>
<td>3</td>
<td>0.010</td>
<td>0.5</td>
<td>54.4</td>
<td>52.1</td>
</tr>
</tbody>
</table>

- Results for higher link capacity are more accurate.
Results: CUBIC and Reno connections, Average window size

Figure: Avg. window for CUBIC and Reno connections sharing a link with $C = 10$Mbps, $p_1 = p_2 = 0.005$
**Results:** CUBIC and Reno connections, Throughput

**Figure:** Throughput for CUBIC and Reno connections sharing a link with $C = 10\text{Mbps}$, $p_1 = p_2 = 0.005$
Conclusions and Future Work

- Built a model for multiple TCP CUBIC connections.
- Validated our results through NS simulations.
- Considered a single bottleneck queue. Multiple bottleneck queues would be an interesting extension.
Thank You!