TCP Congestion Control: Algorithms and Analysis

Simon S. Lam
Department of Computer Sciences
The University of Texas at Austin

Little’s Law

\[
\text{average population} = \frac{\text{average delay} \times \text{throughput}}{	ext{(average delay) \times (throughput)}}
\]

\[
\text{average delay} = \frac{1}{N} \sum_{i=1}^{N} \text{delay}_i
\]

where \(N\) is number of departures

throughput = \(\frac{N}{T}\)

where \(T\) is duration of observation

average population (to be defined)

Try homework problem at
http://www.cs.utexas.edu/users/lam/cs356/homework/hw2.html
Consider an infinite array, Source, at the sender, and an infinite array, Sink, at the receiver.

**Sliding Window Protocol**

- The send window is the range of the source array that the sender is allowed to transmit from, denoted as $[0, s]$.
- The receive window is the range of the sink array that the receiver is allowed to acknowledge, denoted as $[r, r + RW - 1]$.

The average population of the system can be calculated as:

$$\text{average population} = \frac{1}{T} \int_0^T n(t) dt$$
Sliding Windows in Action

- Data unit $r$ has just been received by P2
  - Receive window slides forward
- P2 sends cumulative ack with sequence number it expects to receive next ($r+3$)

```
<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Send Window</th>
<th>Acknowledged</th>
<th>Unacknowledged</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a-1</td>
<td>s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Source: P1 (Sender)

```
<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Send Window</th>
<th>Acknowledged</th>
<th>Unacknowledged</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Sink: P2 (Receiver)

```
<table>
<thead>
<tr>
<th>Sequence Number</th>
<th>Send Window</th>
<th>Acknowledged</th>
<th>Unacknowledged</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

TCP Congestion Control (Simon Lam)
Window Flow Control

- ~ $W$ packets per RTT when no loss
- Lost packet detected by missing ACK
  (note: timeout value $T_O > RTT$)

TCP Congestion Control (Simon Lam)

Throughput (send rate)

- Limit the number of unacked transmitted packets in the network to window size $W$

- Throughput $\approx \frac{W}{RTT}$ packets/sec
  $\approx \frac{W \times MSS}{RTT}$ bytes/sec

- Where did we apply Little’s Law?
  Answer: Consider send buffer

TCP Congestion Control (Simon Lam)
Clarifications

- Average number in the send buffer is typically less than $W$ unless packet arrival rate to send buffer is infinite → previous formula provides a throughput upper bound.

- If each packet may be lost with rate $p$, then the average delay is

$$d = (1 - p) \times RTT + p \times T_o$$

Since $T_o > RTT$, actual throughput is smaller.

- With loss, goodput is

$$g = (1 - p) \times \text{throughput}$$

Note: in some papers and other context (e.g., random access protocols), goodput is called throughput. To avoid confusion, throughput is called send rate.

Effect of Congestion

- $W$ too big for each of many flows → congestion

- Packet loss → transmissions on links prior to packet loss are wasted

- Congestion collapse due too many retransmissions and too much waste

- October 1986, Internet had its first congestion collapse
TCP Window Control

- **Receiver flow control**
  - Avoid overloading receiver
  - \( rwnd \): receiver (advertised) window
  - Receiver sends \( rwnd \) to sender

- **Network congestion control**
  - Sender tries to avoid overloading network
  - It infers available network capacity from "loss indications"
  - \( cwnd \): congestion window

- **Sender sets** \( W = \min (cwnd, rwnd) \)

Receiver Flow Control

- Receiver advertises \( rwnd \) with each packet it sends
- Size of \( rwnd \) indicates available space in receive buffer
  - decreased when data is received from IP layer and ack'd
  - increased when data is consumed by application process
Network Congestion Control

- Sender calculates $cwnd$ from indications of network congestion
- Congestion indications
  - timeout (loss)
  - dupACK (loss likely)
  - queueing delay
  - mark (needs ECN)
- TCP algorithms to calculate $cwnd$
  - Tahoe, Reno, Vegas, ...
- Link algorithms:
  - RED, REM ...

TCP & AQM

Congestion measures $p_f(t)$ for distributed feedback control of $x_i(t)$
- loss and dupACK (DropTail)
- queueing delay (Vegas)

with the help of active queue management (AQM)
- queue length (RED)
- price (REM)
TCP Congestion Control

- Tahoe (Jacobson 1988)
  - Slow Start
  - Congestion Avoidance
  - Fast Retransmit
- Reno (Jacobson 1990)
  - Fast Recovery
  - Its variants: NewReno, SACK
- Vegas (Brakmo & Peterson 1994)
  - New Congestion Avoidance
- AQM
  - RED (Floyd & Jacobson 1993)
    - Probabilistic marking or dropping
  - REM (Athuraliya & Low 2000)
    - Clear buffer, match rate
- Others...

Slow Start

- Start with \( cwnd = 1 \)
- On each successful ACK, increment \( cwnd \)
  \[ cwnd \leftarrow cwnd + 1 \]
- Exponential growth of \( cwnd \)
  
  \[ cwnd \leftarrow 2 \times cwnd \]

- Enter CA when \( cwnd \geq ssthresh \)

- For initial slow start, ssthresh is set to a very large value (e.g., 65 Kbytes)

Note: for clarity, \( cwnd, rwnd, \) and \( ssthresh \) are counted in packets (segments) rather than in bytes
**Slow Start**

- **sender**
  - cwnd
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 1 RTT
- **receiver**
  - data packet
  - ACK
- $\text{cwnd} \leftarrow \text{cwnd} + 1$ (for each ACK)

**Congestion Avoidance**

- **CA starts when**
  - $\text{cwnd} \geq \text{ssthresh}$
- **On each successful ACK:**
  - $\text{cwnd} \leftarrow \text{cwnd} + 1/\text{cwnd}$
- **Linear growth of cwnd**
  - Each RTT:
    - $\text{cwnd} \leftarrow \text{cwnd} + 1$
**Packet Loss**

- **Assumption:** loss indicates congestion
- Packet loss detected by
  - Retransmission timeout (RTO timer)
  - Duplicate ACKs (at least 3)

<table>
<thead>
<tr>
<th>Packets</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>

| Acknowledgements | 1 | 2 | 3 | 3 | 3 | 3 |

**Fast Retransmit**

- A timeout is quite long (> RTT)
- Upon receiving 3 dupACKs, immediately retransmit without waiting for timeout

- Adjusts ssthresh
  \[ ssthresh \leftarrow \max\left(\frac{\text{flightsize}}{2}, 2\right) \]
  where flightsize is number of outstanding packets, which may be less than \( W = \min(\text{rwnd}, \text{cwnd}) \)

- Enter Slow Start (\( \text{cwnd} = 1 \))
**TCP Tahoe** (Jacobson 1988)

- **cwnd**

**SS: Slow Start**
**CA: Congestion Avoidance**

**Successive Timeouts**
- When there is another timeout, double the timeout value.
- Keep doing so for each additional loss-retransmission.
  - Exponential backoff up to max timeout value equal to 64 times initial timeout value.

Note: red line in figure denotes a loss indication.
Summary: Tahoe

- **Basic ideas**
  - Probe network for spare capacity during SS and CA and increase send rate
  - Drastically reduce rate on congestion indication
  - Self-clocking
  - Error recovery by retransmission
  - Round trip time estimation (to get $T_0$ value)

```plaintext
for every ACK {
    if (W < ssthresh) then W++ (SS)
    else W += 1/W (CA)
}

for every loss indication {
    ssthresh = W/2
    W = 1
}
```

TCP Tahoe (Jacobson 1988)

**TCP Tahoe (Jacobson 1988)**

![Graph showing TCP Tahoe](image)

- **SS**: Slow Start
- **CA**: Congestion Avoidance
**TCP Reno (Jacobson 1990)**

- **SS**: Slow Start
- **CA**: Congestion Avoidance
- Fast retransmission/fast recovery

**TCP Reno (another scenario)**

- 3 dupACKs
- Initial slow start
- Slow start until cwnd reaches ssthresh
- 3 dupACKs
- cwnd halved
- Slow start until cwnd reaches ssthresh

TCP Congestion Control (Simon Lam)
**Fast recovery (in detail)**

- **Idea:** Each dupACK represents a packet successfully received. Therefore, no need for very drastic action.
- **Enter FR/FR after 3 dupACKs**
  - Set $ssthresh \leftarrow \max(flightsize/2, 2)$
  - Retransmit lost packet
  - Set $cwnd \leftarrow ssthresh + \#\text{dupACKs}$ (*window inflation*)
  - Wait till $W = \min(rwnd, cwnd)$ is large enough; transmit new packet(s)
  - On non-dup ACK (1 RTT later), set $cwnd \leftarrow ssthresh$ (*window deflation*)
- **Enter CA**

---

**Example: FR/FR**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- **cwnd**: 8
- **ssthresh**: 4
- **Exit FR/FR**: 8

- **Above scenario:** Packet 1 is lost, packets 2, 3, and 4 are received; dupACKs with seq. no. 0 returned.
- **Fast retransmit**
  - Retransmit on 3 dupACKs
- **Fast recovery**
  - Inflate window such that new packets 9, 10, and 11 can be sent while repairing loss.
Summary: Reno

- Basic ideas
  - Fast recovery avoids slow start
  - dupACKs: fast retransmit + fast recovery
  - Timeout: fast retransmit + slow start

AIMD in steady state

**additive increase:**
- Increase cwnd by 1 MSS every RTT in the absence of any loss event

**multiplicative decrease:**
- Cut cwnd in half after 3 dupACKs
**TCP throughput (send rate)**

- We derived the approximate formula

\[
\text{throughput} = \frac{W}{RTT} \text{ packets/sec}
\]

- \( W \) changes with the arrival of each congestion indication

- To calculate (average) send rate, we need the average value of \( W \)

Q: \( W \) is a function of what parameter?

---

**First approximation**


- No slow-start, no timeout, long-lived TCP connection

- Independent identically distributed "periods"

- Each packet may be lost with probability \( p \)
Geometric Distribution

Ave. no. of transmissions to get first loss

\[ \bar{n} = \sum_{i=1}^{\infty} ib_i = \sum_{i=1}^{\infty} i(1-p)^{i-1} p \]

\[ = p \sum_{i=1}^{\infty} i(1-p)^{i-1} \]

\[ = -p \frac{d}{dp} \sum_{i=1}^{\infty} (1-p)^i = -p \frac{d}{dp} \sum_{i=0}^{\infty} (1-p)^i \]

\[ = -p \frac{d}{dp} \frac{1}{1-1+p} = p \frac{1}{p^2} \]

\[ = \frac{1}{p} \]

Similarly, ave. no. of transmissions to get first success is \( 1/(1-p) \)

First approximation (cont.)

- Average number of packets delivered in one period (area under one saw-tooth)

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

\[ \frac{W^2}{2} + \frac{1}{2} \left( \frac{W}{2} \right)^2 = \frac{3}{8} W^2 \]

- Average number of packets sent per period (incl. loss at the end) is \( 1/p \)

- Equating the two and solving for \( W \), we get

\[ \frac{3}{8} W^2 \]

\[ = \frac{1}{p} \]

\[ RTT \left( \frac{2}{\sqrt{3p}} \right) = \frac{1}{RTT} \sqrt{\frac{3}{2p}} \]
**TCP ACK generation** [RFC 1122, RFC 2581]

<table>
<thead>
<tr>
<th>Event at Receiver</th>
<th>TCP Receiver action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed</td>
<td>Delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK</td>
</tr>
<tr>
<td>Arrival of in-order segment with expected seq #. One other segment has ACK pending</td>
<td>Immediately send single cumulative ACK, ACKing both in-order segments</td>
</tr>
<tr>
<td>Arrival of out-of-order segment higher-than-expect seq. # . Gap detected</td>
<td>Immediately send duplicate ACK, indicating seq. # of next expected byte</td>
</tr>
<tr>
<td>Arrival of segment that partially or completely fills gap</td>
<td>Immediate send ACK, provided that segment starts at lower end of gap</td>
</tr>
</tbody>
</table>

**Receiver implements Delayed ACKs**

- Receiver sends one ACK for every two packets received -> each saw-tooth is $W \times RTT$ wide
  -> area under a saw-tooth is $\frac{3W^2}{4}$
- Send rate is $\frac{1}{RTT} \sqrt{\frac{3}{4p}}$
- One ACK for every $b$ packets received -> send rate is $\frac{1}{RTT} \sqrt{\frac{3}{2bp}}$

Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose

**Motivation**

- Previous formulas not so accurate when loss rates are high
- TCP traces show that there are more loss indications due to timeouts (TO) than due to triple dupACKs (TD)
Objectives

- More accurate steady-state throughput formula as a function of loss rate and RTT by also accounting for TO behavior of a TCP connection
- Formula applicable over a wider range of loss rates
- Explicit statements of assumptions and approximations used in derivation of throughput formula
- Formula to include the impact of a small rwnd

Many assumptions and approximations

- A1. TCP sender is saturated, i.e., source application process always has a packet to send when send window has space available
  - i.e., bulk transfer application
- A2. Slow Start not modeled
- A3. Time to send all packets in a window is smaller than RTT
  - i.e., transmission rate is not too low
A3. Time to send $W$ packets is less than RTT

- ACK reception marks the end of current round and beginning of next round.
- Approximation: For $b > 1$, ACK is not received immediately after one RTT, but it is so assumed in the analysis.

AIMD evolution of Window Size over time

- Each TD period is ended by a TD loss indication.
- TDP$_1$ period has duration $A_1$ rounds
- A4. Duration of a round (RTT) is independent of window size
  - approximation (poor for a slow line)
- A5. No window inflation in Fast Recovery
  - approximation
Markov regenerative assumption

- For the \(i\)-th TD period, \(W_i\) is window size at the end of the period, \(Y_i\) is the number of packets sent in the period.
- A6. Assume \(\{W_i\}\) to be a Markov regenerative process with rewards \(\{Y_i\}\).
- Given A6, the steady-state TCP throughput is

\[
B = \lim_{t \to \infty} B_t = \lim_{t \to \infty} \frac{N_t}{t} = \frac{E[Y_t]}{E[A_t]} \triangleq \frac{E[Y]}{E[A]}
\]

Consider \(i\)-th TD period

- One ACK after receiving \(b\) packets (\(b = 2\) in above figure) -> linear increase has a slope of \(1/b\) packet per RTT.
- Number of rounds is \(X_i + 1\).
- \(\alpha_i\) is the first packet lost in \(i\)-th TD period.
Loss assumptions

- **A7.** Losses in different rounds are independent
  - approximation

- **A8.** Losses within the same round are correlated as follows: If a packet is lost, all remaining packets transmitted until the end of that round are also lost
  - approximation - bursty loss behavior but only within the same round
  - all lost packets in the same round are counted as a single loss indication when estimating $p$

\[ E[\alpha] = 1/p \]
\[ E[r] = RTT \]
\[ E[Y] = E[\alpha] + E[W] - 1 = \frac{1}{p} - 1 + E[W] \]

From $W_i = \frac{W_i}{2} + \frac{X_i}{b}$, we have
\[ E[X] = \frac{b}{2} E[W] \]
\[ E[A] = (E[X] + 1)E[r] = \left(\frac{b}{2} E[W] + 1\right)RTT \]

Send rate $B = \frac{E[Y]}{E[A]} = \frac{\frac{1}{p} - 1 + E[W]}{\left(\frac{b}{2} E[W] + 1\right)RTT}$

TCP Congestion Control (Simon Lam) 45
**AIMD throughput derivation (2)**

Another way to compute $E[Y]$

\[ Y_i = \sum_{k=0}^{X_i/b-1} \left( \frac{W_{i+1}}{2} + k \right) b + \beta_i \]

\[ = \frac{X_i W_{i+1}}{2} + \frac{X_i}{2} \left( \frac{1}{b} - 1 \right) + \beta_i \]

\[ = \frac{X_i}{2} (W_i + \frac{W_{i+1}}{2} - 1) + \beta_i \]

Let $E[\beta]$ be $E[W]$ and we have

\[ E[Y] = \frac{E[X]}{2} \left( E[W] + \frac{E[W]}{2} - 1 \right) + E[\beta] \]

\[ = \frac{bE[W]}{4} \left( E[W] + \frac{E[W]}{2} - 1 \right) + \frac{E[W]}{2} \]

\[ \text{\textbullet\quad A9. Assume that } \{X_i\} \text{ and } \{W_i\} \text{ are mutually independent i.i.d. sequences of random variables} \]

**AIMD throughput (3)**

- Equate the two previous formulas for $E[Y]$. Solve the quadratic equation with $E[W]$ as the only unknown

\[ E[Y] = \frac{1}{p} - 1 + E[W] \]

\[ = \frac{bE[W]}{4} \left( E[W] + \frac{E[W]}{2} - 1 \right) + \frac{E[W]}{2} \]

\[ E[W] = \frac{2 + b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left( \frac{2 + b}{3b} \right)^2} \]

Send rate \[ B(p) = \frac{\frac{1-p}{p} + E[W]}{E[A]} \]

TCP Congestion Control (Simon Lam)
**AIMD throughput (4)**

To get a simple formula, collect terms that are $o(1/\sqrt{p})$

$$E[W] = \frac{8}{\sqrt{3bp}} + o(1/\sqrt{p})$$

$$E[X] = \frac{b}{2} E[W] = \frac{2b}{\sqrt{3p}} + o(1/\sqrt{p})$$

Send rate $B(p) = \frac{1/p + o(1/p)}{RTT} \approx \frac{1}{RTT} \sqrt{\frac{3}{2bp}} + o(1/\sqrt{p})$

**AIMD with TO**

- Let $n_i$ denote the number of TD periods within a cycle ending in $i$-th TO period, $R_i$ denote no. of retransmissions in $i$-th TO period
- A10. $\{n_i\}$ form an i.i.d. sequence, independent of $\{Y_{ij}\}$ and $\{A_{ij}\}$

$$M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i, \quad S_i = \sum_{j=1}^{n_i} A_{ij} + Z_i^{TO}$$
Throughput of AIMD with TO (1)


\[ E[S] = E[n]E[A] + E[Z^{TO}] \]

Send rate \( B = \frac{E[M]}{E[S]} = \frac{E[n]E[Y] + E[R]}{E[n]E[A] + E[Z^{TO}]} \)

\[ B = \frac{E[Y] + Q \times E[R]}{E[A] + Q \times E[Z^{TO}]} \]

where \( Q \triangleq \frac{1}{E[n]} \)

\[ E[R] = \frac{1}{1 - p} \]

with \( Q \) and \( E[Z^{TO}] \) to be determined

Assumption of Markov regenerative process again.

\[ \leftarrow \text{Probability that a given loss indication is a TO} \]

TCP Congestion Control (Simon Lam)

51

Approximate solution for Q

A given loss indication is a TO is the union of two events \( \Leftrightarrow \) Two or less acked packets in penultimate round or two or less acked packets in final round

TCP Congestion Control (Simon Lam)

52
Approximate solution for $Q$ (cont.)

$$A(w,k) = \frac{(1-p)^k p}{1-(1-p)^w}$$

\[-\text{penultimate round of } w \text{ packets, first } k \text{ packets ack'd given there is a loss}\]

$$C(k,m) = (1-p)^m p, \quad m \leq k-1$$

$$C(k,m) = (1-p)^m, \quad m = k$$

\[-\text{for last round, } k \text{ packets sent, } m \text{ packets ack'd in sequence}\]

$$\hat{Q}(w) = 1 \quad \text{if } w \leq 3$$

\[-\text{at most } 2 \text{ dupACKs}\]

$$\hat{Q}(w) = \sum_{k=0}^{3} A(w,k) + \sum_{k=3}^{\infty} A(w,k) \sum_{m=0}^{2} C(k,m)$$

\[-\text{probability of fewer than } 3 \text{ packets sent successfully in penultimate round or less than } 3 \text{ acks in last round}\]

TCP Congestion Control (Simon Lam) 54

\---

Approximate solution for $Q$ (cont.)

After algebraic manipulations, we have

$$\hat{Q}(w) = \min\left(1, \frac{(1 - (1-p)^3)(1 + (1-p)^3(1-(1-p)^{w-3}))}{1-(1-p)^w}\right)$$

Observe (for example, using L'Hôpital's rule) that

$$\lim_{p \to 0} \hat{Q}(w) = \frac{3}{w}.$$ 

Numerically we find that a very good approximation of $\hat{Q}$ is

$$\hat{Q}(w) \approx \min(1, \frac{3}{w})$$

- $Q$ is $E[\hat{Q}(w)]$
- But we don’t know the probability distribution of $W_i$
- Approximation $Q = \hat{Q}(E[W]) \approx \min(1, \frac{3}{E[W]} \approx \min(1, \frac{3bp}{8})$

TCP Congestion Control (Simon Lam) 54
Throughput of AIMD with TO (2)

\[ P[R = k] = p^{k-1}(1 - p) \quad \text{for } k = 1, 2, \ldots \]
\[ L_k = (2^k - 1)T_0 \quad \text{for } k \leq 6 \]
\[ = (63 + 64(k - 6))T_0 \quad \text{for } k \geq 7 \]
\[ E[Z^{70}] = T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1 - p} \]
\[ \Delta = T_0 \frac{f(p)}{1 - p} \approx T_0 (1 + 32p^5) \]

send rate
\[ B(p) = \frac{E[Y] + Q \times E[R]}{E[A] + Q \times E[Z^{70}]} \]
\[ B(p) = \frac{1 - p}{p} + E[W] + \hat{Q}(E[W]) \frac{1}{1 - p} \]
\[ \frac{RTT(E[X]+1) + \hat{Q}(E[W])T_0}{RTT(E[X]+1) + \hat{Q}(E[W])T_0} \frac{f(p)}{1 - p} \]

TCP Congestion Control (Simon Lam) 55

Throughput of AIMD with TO (3)

\[ B(p) = \frac{1 - p}{p} + E[W] + \hat{Q}(E[W]) \frac{1}{1 - p} \]
\[ \frac{1}{RTT(E[X]+1) + \hat{Q}(E[W])T_0} \frac{f(p)}{1 - p} \]
\[ = \frac{1}{RTT \left( \frac{2b}{\sqrt{3p}} \right) + \min \left( \frac{3bp}{8}, 1.3 \sqrt{\frac{3bp}{8}} \right)} (1 + 32p^5)T_0 \]
\[ = \frac{1}{RTT \left( \frac{2bp}{3} \right) + \min \left( \frac{3bp}{8}, 1.3 \sqrt{\frac{3bp}{8}} \right)} p(1 + 32p^5)T_0 \]

TCP Congestion Control (Simon Lam) 56
**Impact of receiver’s rwnd limitation**

Compute $E[W]$. If $E[W] < W_{\text{max}}$, use Eq. (27):

$$B(p) \approx \frac{1-p}{p} + E[W] + \hat{Q}(E[W]) \cdot \frac{1}{1-p}$$

$$\frac{RTT(E[X]+1)+\hat{Q}(E[W])T_0}{RTT(E[X]+1)+\hat{Q}(E[W])T_0} \cdot f(p)$$

if $E[W] < W_{\text{max}}$.

Otherwise, use $W_{\text{max}}$ for $E[W]$ and recompute $E[X]$ (derivation omitted).

**Full model Eq. (31)**

**Impact of receiver’s rwnd limitation—approximate model**

Use the well-known Eq. (29) from before,

$$B(p) \approx \min\left(\frac{W_{\text{max}}}{RTT}, \frac{1}{RTT\left(\sqrt{\frac{2bp}{3}}\right) + \min\left(1,\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)T_0}\right)$$

which is referred to as Eq. (32)
Summary data from traces (1 hour)

- Saturated TCP sender
- $p$ computed from dividing total no. of loss indications by total number of packets sent
- $RTT$ and $T_D$ values are averaged over entire 1-hour trace

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
<th>Packets Sent</th>
<th>Loss Indic.</th>
<th>$T_D$</th>
<th>$T_0$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>$T_4$</th>
<th>$T_5$ or larger</th>
<th>$RTT$</th>
<th>Time Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>macc</td>
<td>alps</td>
<td>54402</td>
<td>7 19 65 67</td>
<td>15</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0.039 3 5.995</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>kibbles</td>
<td>58236</td>
<td>725 706 411</td>
<td>17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.343 2.495</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>green</td>
<td>38504</td>
<td>343 372 444</td>
<td>22</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.226 3.405</td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>martha</td>
<td>50383</td>
<td>494 2 474 17</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.283 2.146</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>nessie</td>
<td>67932</td>
<td>849 1 654 35</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.185 2.416</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>spell</td>
<td>11791</td>
<td>784 47 702 56</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.218 2.274</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>connie</td>
<td>81113</td>
<td>158 948 597 44</td>
<td>17</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.204 2.159</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>macc</td>
<td>town</td>
<td>7938</td>
<td>281 1 199 37 18</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>0.276 3.397</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>alps</td>
<td>57177</td>
<td>818 7 288 164 56 17</td>
<td>4</td>
<td>2</td>
<td>0.872 3.489</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>kibbles</td>
<td>32043</td>
<td>831 390 685 67 32 2</td>
<td>0</td>
<td>0</td>
<td>0.482 3.304</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>green</td>
<td>67770</td>
<td>1142 414 582 79 20 9</td>
<td>4</td>
<td>2</td>
<td>0.254 0.837</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>martha</td>
<td>96856</td>
<td>1051 33 1444 197 54 13</td>
<td>5</td>
<td>3</td>
<td>0.423 0.617</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>spell</td>
<td>67350</td>
<td>671 72 539 56 4 0</td>
<td>0</td>
<td>0</td>
<td>0.538 0.749</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>connie</td>
<td>73486</td>
<td>1081 658 865 132 45 14</td>
<td>9</td>
<td>1</td>
<td>0.213 0.802</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>void</td>
<td>town</td>
<td>3365</td>
<td>816 7 344 209 100 35 27 12</td>
<td>0.272 1.306</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>alps</td>
<td>13460</td>
<td>1466 0 1068 247 87 33 18 8</td>
<td>0 0 0.104 1.359</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>kibbles</td>
<td>62237</td>
<td>1783 197 1467 76 10 3 0</td>
<td>0 0 0.255 0.429</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>green</td>
<td>80673</td>
<td>2123 498 1650 88 2 1</td>
<td>0 0 0.298 0.508</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>spell</td>
<td>55087</td>
<td>1120 0 539 337 56 2 1</td>
<td>0 0 0.338 0.953</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>connie</td>
<td>88406</td>
<td>2320 685 1444 142 38 9 4</td>
<td>0 0 0.210 0.705</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>labol</td>
<td>town</td>
<td>89944</td>
<td>1388 1 1164 318 17 7 5 3 0.150 0.530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ptls</td>
<td>alps</td>
<td>89971</td>
<td>781 0 577 111 68 18 8 2 0.068 0.728</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ptls</td>
<td>kibbles</td>
<td>44801</td>
<td>1186 15 1564 186 53 12 5 0 0.226 0.700</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ptls</td>
<td>green</td>
<td>14351</td>
<td>1422 43 944 272 191 34 14 6 0.275 1.454</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Summary data from 100s traces

- Each row represents results of 100 traces each of 100 seconds in duration for same S-D pair
- Totals are cumulative over 100 traces
- $RTT$ and $T_D$ are average values over 100 traces for same S-D pair

TCP Congestion Control (Simon Lam) 59
Experimental comparison (1)

- Each point represents number of packets in 100s interval of trace
- T0 ~ single TO, T1 ~ at least 1 double TO in trace, etc.
- "TD Only" is analytic model by Mathis et al.
- Note: $W_{max}$ is only 6 in Figure 7

Experimental comparison (2)

$W_{max} = 33$

$W_{max} = 44$
Experimental comparison (3)

**Figure 11:** void to tove 

$W_{\text{max}}=8$

**Figure 12:** babel to alps 

$W_{\text{max}}=48$

TCP Congestion Control (Simon Lam) 63

Accuracy of approximate model

**Figure 18:** manic to spiff, with predictions by both full and approximate models 

$(W_{\text{max}}=32)$

TCP Congestion Control (Simon Lam) 64
Average errors

\[
\text{ave. error} = \frac{\sum_{\text{observations}} \left| N_{\text{predicted}} - N_{\text{observed}} \right|}{\text{no. of observations}}
\]

Figure 19: Comparison of the models for 1hr traces

Figure 20: Comparison of the models for 100s traces

Conclusions

- A much more rigorous analysis than the one by Mathis et al.
- Numerous assumptions and approximations used but (almost) all of them are explicitly stated
- Large amount of experimental measurements on the Internet to validate accuracy of the full model (less for the approximate model)
- Throughput formula accounts for loss indications due to TO as well as rwnd restriction
  - Using the formula requires accurate measurements of loss rate and RTT values (which could be tricky)
  - For TCP Reno and drop-tail router
- Accuracy (like beauty) is in the eye of the beholder. What do you think?
TCP Throughput limited by loss rate

- TCP average throughput (approximate) in terms of loss rate, $L$:
  \[
  \frac{1.22 \cdot MSS}{RTT \sqrt{p}}
  \]

- Example: 1500-byte segments, 100ms RTT, to get 10 Gbps throughput, loss rate needs to be very low
  \[p = 2 \times 10^{-10}\]

- New version of TCP needed for connections with high-delay bandwidth product
  - addressed in paper by Katabi’s et al

---

The End