TCP Congestion Control: Algorithms and Analysis

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

Little's Law

Average population = (average delay) x (throughput)

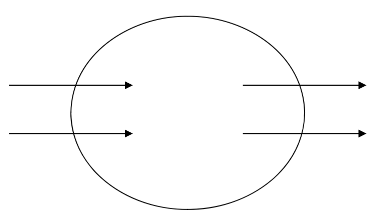
$$\text{average delay} = \frac{1}{N} \sum_{i=1}^{N} \text{delay}_i$$

where N is number of departures

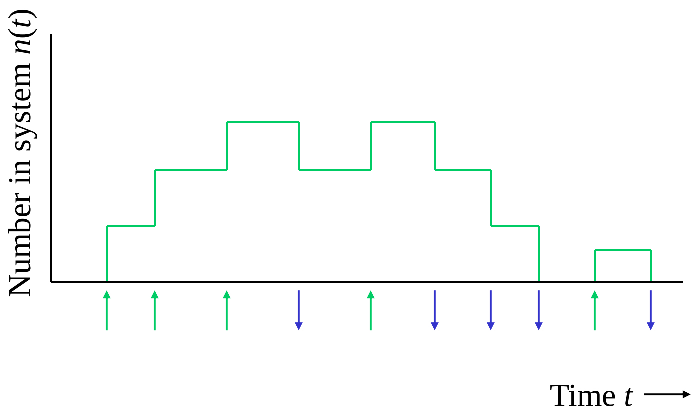
$$\text{throughput} = N/T$$

where T is duration of observation

average population (to be defined)



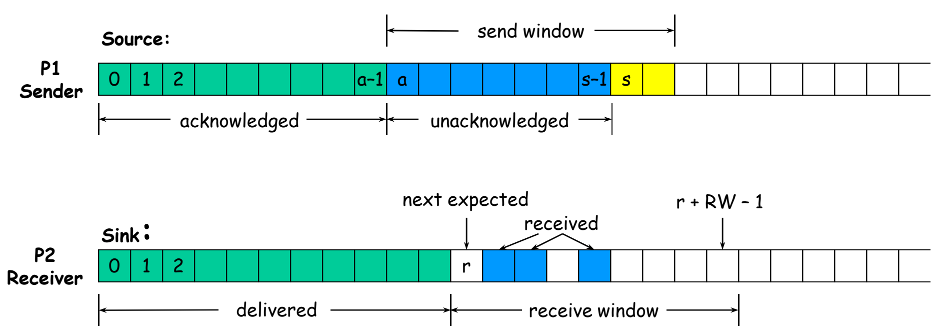
Try homework problem at
http://www.cs.utexas.edu/users/lam/cs356/homework/hw2.html



average population $= \frac{1}{T} \int_0^T n(t)dt$

Sliding Window Protocol

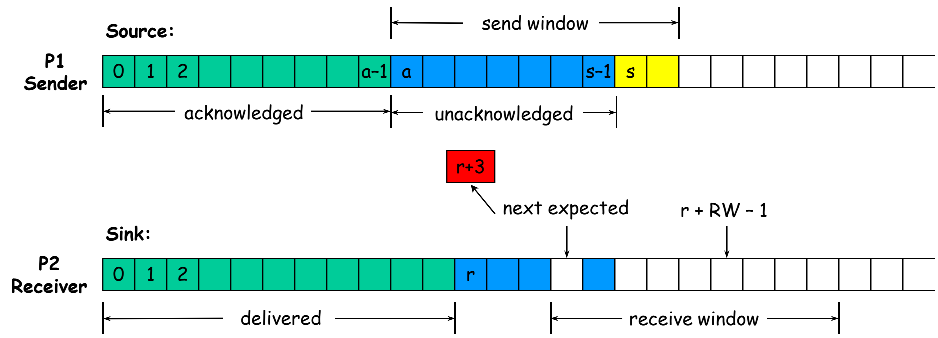
- Consider an infinite array, Source, at the sender, and an infinite array, Sink, at the receiver.



RW receive window size
SW send window size ($s - a \leq SW$)

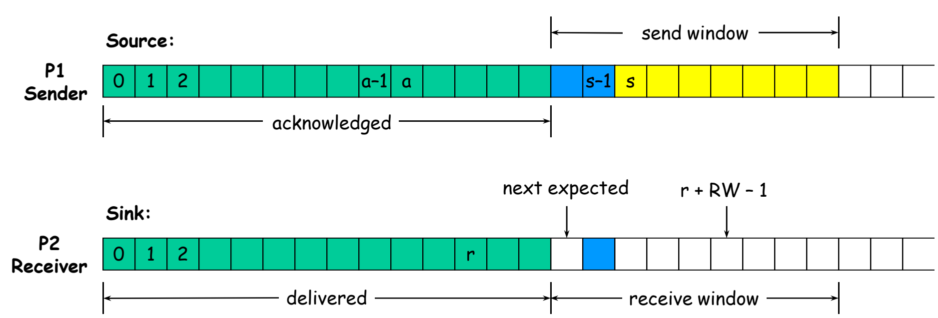
Sliding Windows in Action

- 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)

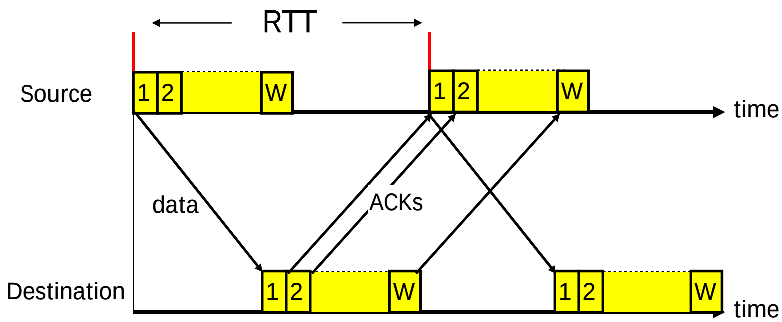


Sliding Windows in Action

- P1 has just received cumulative ack with r+3 as next expected sequence number
  - Send window slides forward



Window Flow Control



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

Throughput (send rate)

- Limit the number of unacked transmitted packets in the network to window size W
- Throughput $\cong \dfrac{W}{RTT}$ packets/sec

$$= \frac{W \times MSS}{RTT} \text{ bytes/sec}$$

- Where did we apply Little's Law?

  Answer: Consider send buffer

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

$$(1-p)\times RTT + p\times T_O$$

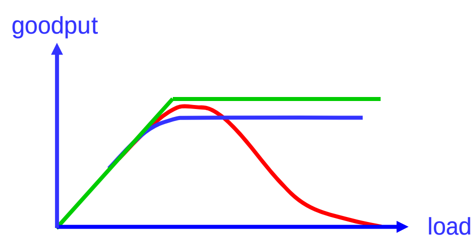
  Since T_O > RTT, actual throughput is smaller.
- With loss, goodput is

$$(1-p)\times throughput$$

  - 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)**

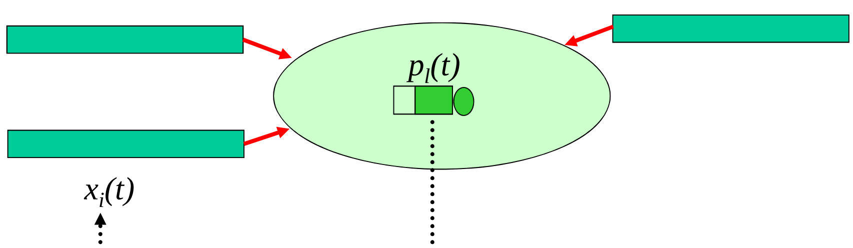
Receiver Flow Control

- 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 $p_l(t)$ 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 ← cwnd + 1
- Exponential growth of cwnd

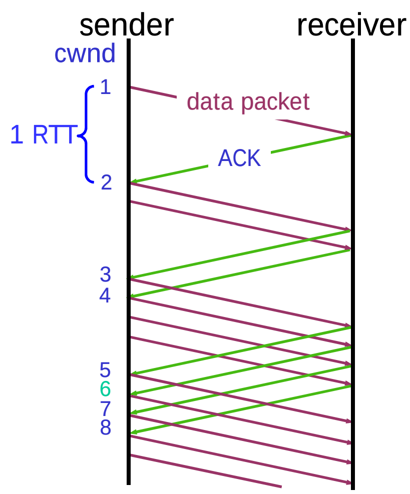
  each RTT: cwnd ← 2 x cwnd

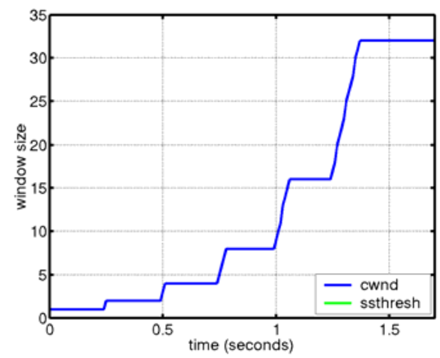
- Enter CA when cwnd >= ssthresh

- 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

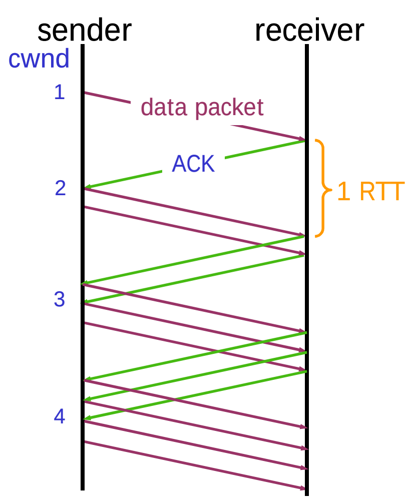
Slow Start





cwnd ← cwnd + 1 (for each ACK)

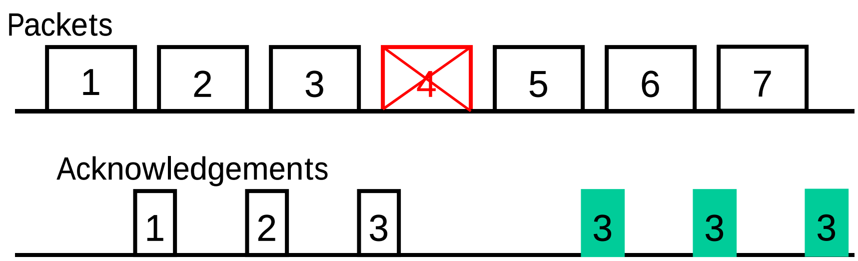
Congestion Avoidance



- CA starts when
  cwnd ≥ ssthresh
- On each successful ACK:
  cwnd ← cwnd + 1/cwnd
- Linear growth of cwnd each RTT:
  cwnd ← cwnd + 1

Packet Loss

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



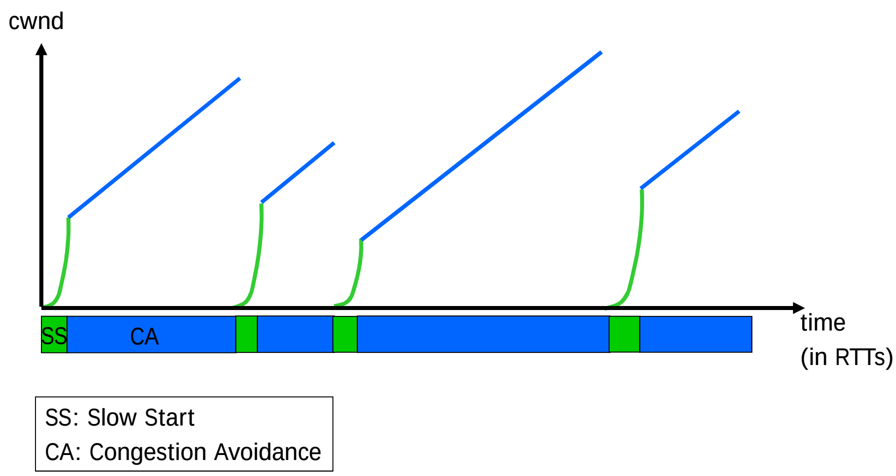
Fast Retransmit

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

  ssthresh ← max(flightsize/2, 2)

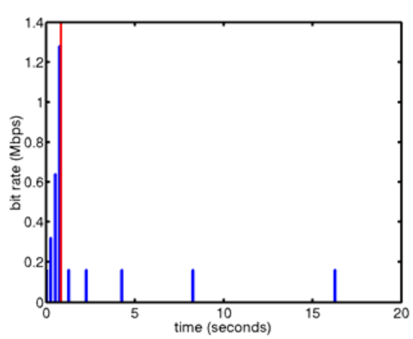
  where flightsize is number of outstanding packets, which may be less than W = min(rwnd, cwnd)
- Enter Slow Start (cwnd = 1)

TCP Tahoe (Jacobson 1988)



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



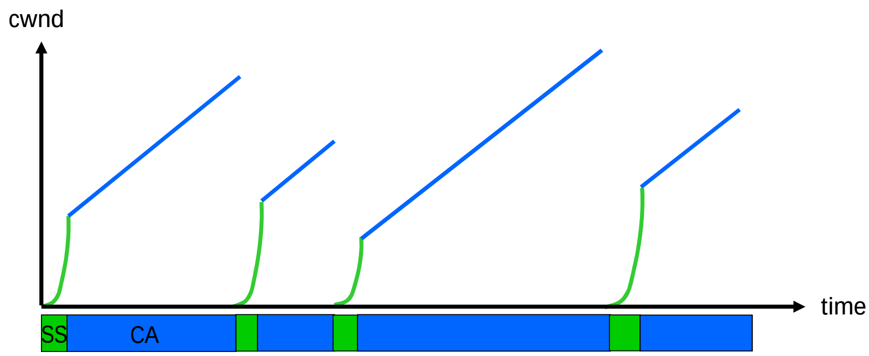
Note: red line in figure denotes a loss indication

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_O value)

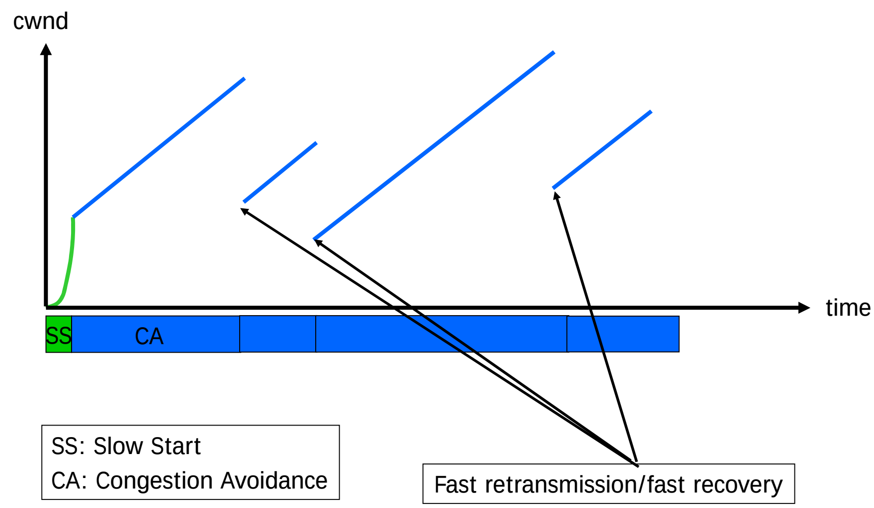
```
for every ACK {
    if (W < ssthresh) then W++     (SS)
    else  W += 1/W                 (CA)
}
for every loss indication {
    ssthresh = W/2
          W  = 1
}
```


TCP Tahoe (Jacobson 1988)



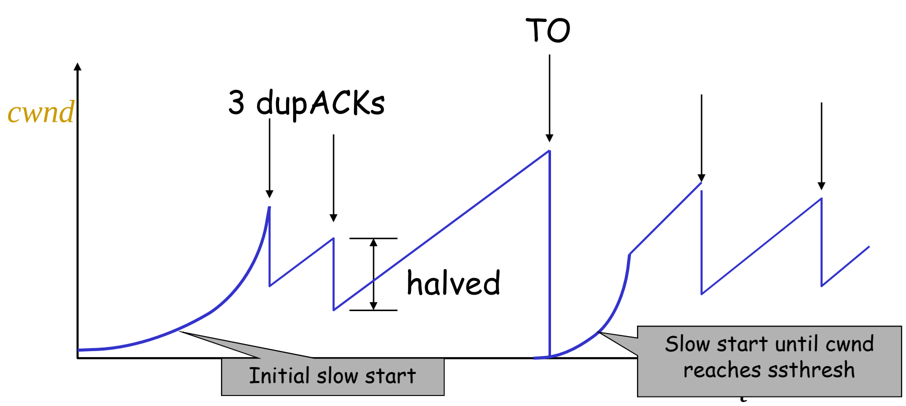
SS: Slow Start
CA: Congestion Avoidance

TCP Reno (Jacobson 1990)



TCP Congestion Control (Simon Lam) 25

TCP Reno (another scenario)

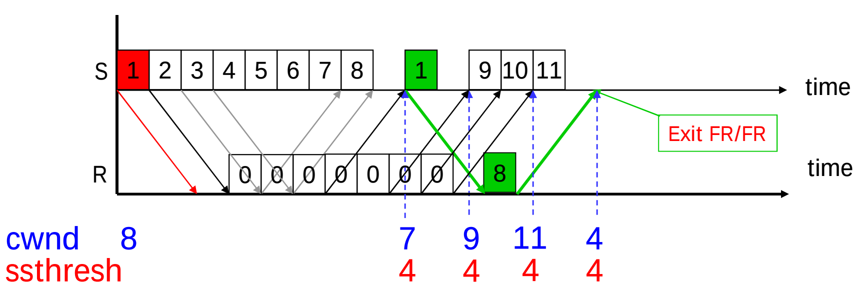


TCP Congestion Control (Simon Lam) 26

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 ← max(flightsize/2, 2)
  - Retransmit lost packet
  - Set cwnd ← ssthresh + #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 ← ssthresh (window deflation)
- Enter CA

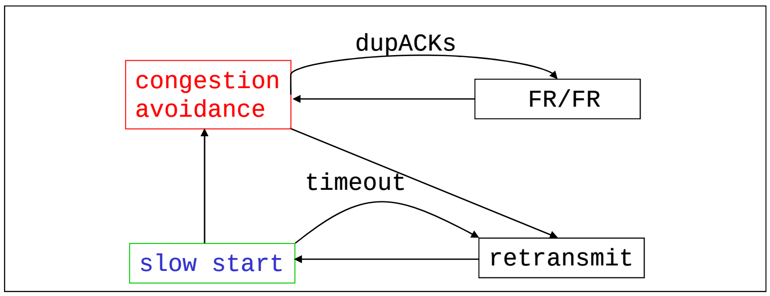
Example: FR/FR



- 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

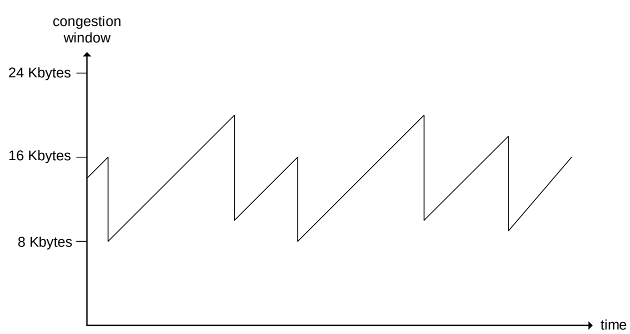
- Basic ideas
  - Fast recovery avoids slow start
  - dupACKs: fast retransmit + fast recovery
  - Timeout: fast retransmit + slow start



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



Long-lived TCP connection

TCP throughput (send rate)

- We derived the approximate formula

throughput = $\frac{W}{RTT}$ packets/sec

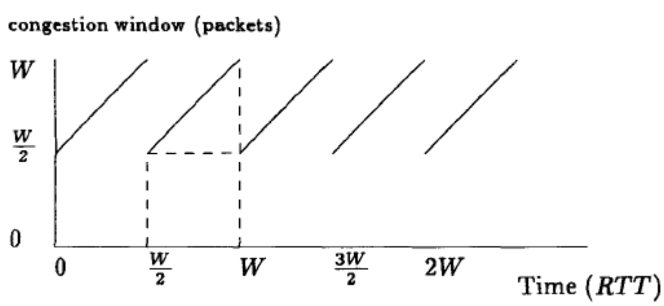
- W changes with the arrival of each congestion indication
- To calculate (average) send rate, we need the average value of W

Q: W is a function of what parameter?

First approximation

M. Mathis, et al., "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *ACM Computer Communicatons Review*, 27(3), 1997.

- 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

$$\begin{aligned}\bar{n} &= \sum_{i=1}^{\infty} i b_i = \sum_{i=1}^{\infty} i(1-p)^{i-1} p \\ &= p \sum_{i=1}^{\infty} i(1-p)^{i-1} \\ &= -p \frac{d}{dp} \sum_{i=1}^{\infty} (1-p)^i = -p \frac{d}{dp} \sum_{i=0}^{\infty} (1-p)^i \\ &= -p \frac{d}{dp} \frac{1}{1-1+p} = p \frac{1}{p^2} \\ &= 1/p\end{aligned}$$

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)

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

- Average number of packets sent per period (incl. loss at the end) is $1/p$
- Equating the two and solving for W, we get

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

send rate (in packets/sec)

$$= \frac{\text{no. of packets/period}}{\text{time per period}} = \frac{\frac{3}{8}W^2}{RTT\left(\frac{W}{2}\right)}$$

$$= \frac{1/p}{RTT\left(\sqrt{\frac{2}{3p}}\right)} = \frac{1}{RTT}\sqrt{\frac{3}{2p}}$$

TCP ACK generation [RFC 1122, RFC 2581]

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

Receiver implements Delayed ACKs

- Receiver sends one ACK for every two packets received -> each saw-tooth is WxRTT 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

$$\frac{1}{RTT}\sqrt{\frac{3}{2bp}}$$

Modeling TCP Throughput: A Simple Model and its Empirical Validation, *Proc. ACM SIGCOMM, 1998*

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)

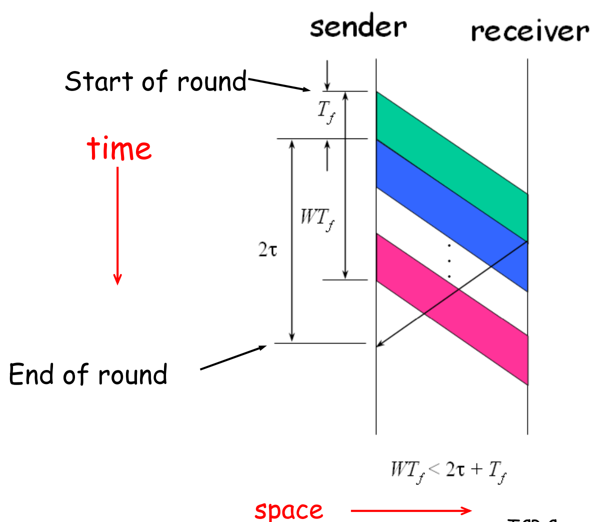
Objectives

- 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

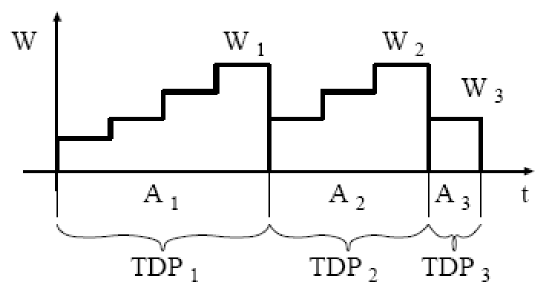
- A1. TCP sender is saturated, i.e., source application process always has a packet to send when send window has space available
  - i.e., bulk transfer application
- A2. Slow Start not modeled
- A3. Time to send all packets in a window is smaller than RTT
  - i.e.,transmission rate is not too low

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



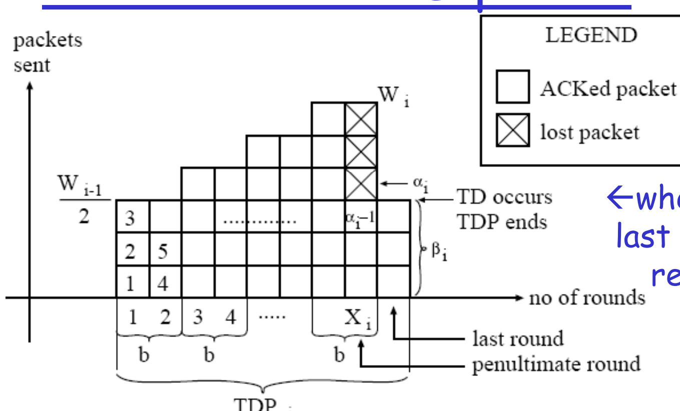
- Each TD period is ended by a TD loss indication.
- TDP_i period has duration A_i rounds
- A4. Duration of a round (RTT) is independent of window size
  - approximation (poor for a slow line)
- A5. No window inflation in Fast Recovery
  - approximation

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

$$B = \lim_{t\to\infty} B_t = \lim_{t\to\infty} \frac{N_t}{t} = \frac{E[Y_i]}{E[A_i]} \triangleq \frac{E[Y]}{E[A]}$$

Consider *i*-th TD period



←when ACK of last packet is received

- 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 $X_i + 1$
- α_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

AIMD throughput derivation (1)

$$E[\alpha] = 1/p$$

$$E[r] = RTT$$

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

From $W_i = \frac{W_{i-1}}{2} + \frac{X_i}{b}$, we have

$$E[X] = \frac{b}{2} E[W]$$

$$E[A] = (E[X]+1)E[r] = (\frac{b}{2} E[W] + 1) RTT$$

<- from A4 that round trip times are independent of W_i

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

α seq. no. of first loss
r round trip time
Y no. of packets sent
W window size
X no. of rounds
A time duration of a period

AIMD throughput derivation (2)

Another way to compute E[Y]

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

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

$$= \frac{X_i}{2}\left(W_i + \frac{W_{i-1}}{2} - 1\right) + \beta_i$$

Let $E[\beta]$ be $E[\frac{W}{2}]$ and we have

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

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

<- A9. Assume that $\{X_i\}$ and $\{W_i\}$ 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

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

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

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

$$\text{send rate} \quad B(p) = \frac{\frac{1-p}{p} + E[W]}{E[A]}$$

AIMD throughput (4)

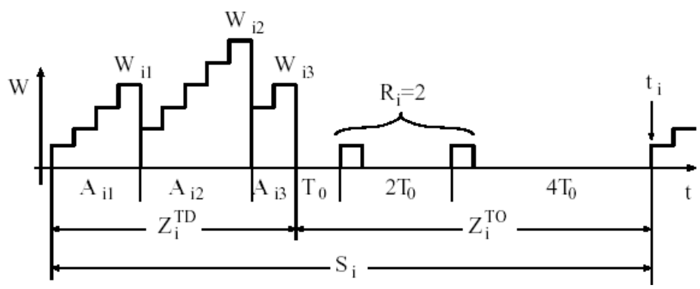
To get a simple formula, collect terms that are o(1/sqrt(p))

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

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

$$\text{send rate } B(p) = \frac{1/p + o(1/p)}{RTT\left(\sqrt{\frac{2b}{3p}} + o(1/\sqrt{p})\right)} \approx \frac{1}{RTT}\sqrt{\frac{3}{2bp}} + o(1/\sqrt{p})$$

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 {Yij} and {Aij}

$$M_i = \sum_{j=1}^{n_i} Y_{ij} + R_i, \quad S_i = \sum_{j=1}^{n_i} A_{ij} + Z_i^{TO}$$

Throughput of AIMD with TO (1)

$E[M] = E[n]E[Y] + E[R]$

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

send rate $B = \dfrac{E[M]}{E[S]} = \dfrac{E[n]E[Y] + E[R]}{E[n]E[A] + E[Z^{TO}]}$

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

where $Q \triangleq \dfrac{1}{E[n]}$

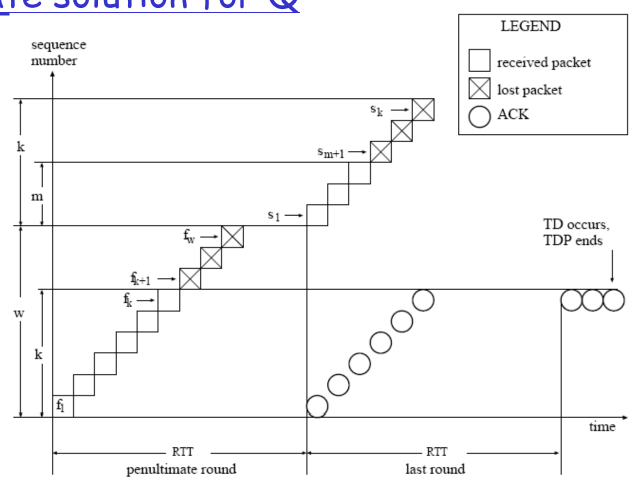
$E[R] = \dfrac{1}{1-p}$

with Q and $E[Z^{TO}]$ to be determined

Assumption of Markov regenerative process again.

<- Probability that a given loss indication is a TO

Approximate solution for Q



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

Approximate solution for Q (cont.)

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

<- penultimate round of w packets, first k packets ack'd given there is a loss

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

<- for last round, k packets sent, m packets ack'd in sequence

$$\hat{Q}(w) = 1 \qquad \text{if } w \le 3$$

<- at most 2 dupACKs

$$= \sum_{k=0}^{2} A(w,k) + \sum_{k=3}^{w} A(w,k) \sum_{m=0}^{2} C(k,m) \qquad \text{if } w \ge 4$$

<- probability of fewer than 3 packets sent successfully in penultimate round or less than 3 acks in last round

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'Hopital'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 $Q \simeq \hat{Q}(E[W]) \simeq \min(1, \frac{3}{E[W]}) \simeq \min(1, 3\sqrt{\frac{3bp}{8}})$

Throughput of AIMD with TO (2)

$$P[R=k]=p^{k-1}(1-p) \quad \text{for } k=1,2,\ldots$$

$$L_k=(2^k-1)T_O \quad \text{for } k\le 6$$

$$=(63+64(k-6))T_O \quad \text{for } k\ge 7$$

<- duration of *k* TOs in a row

$$E[Z^{TO}]=T_O\frac{1+p+2p^2+4p^3+8p^4+16p^5+32p^6}{1-p}$$

$$\triangleq T_O\frac{f(p)}{1-p} \quad \simeq T_0(1+32p^2)$$

<- approximation

$$\text{send rate } B(p)=\frac{E[Y]+Q\times E[R]}{E[A]+Q\times E[Z^{TO}]}$$

$$B(p)\simeq\frac{\frac{1-p}{p}+E[W]+\widehat{Q}(E[W])\frac{1}{1-p}}{RTT(E[X]+1)+\widehat{Q}(E[W])T_O\frac{f(p)}{1-p}}$$

Throughput of AIMD with TO (3)

$$B(p)\simeq\frac{\frac{1-p}{p}+E[W]+\widehat{Q}(E[W])\frac{1}{1-p}}{RTT(E[X]+1)+\widehat{Q}(E[W])T_O\frac{f(p)}{1-p}}$$

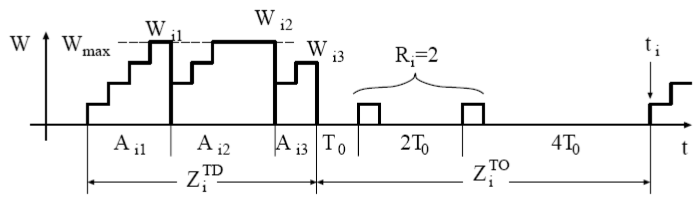
<- Eq. (27) more accurate version of throughput formula

$$\simeq\frac{1/p}{RTT\left(\sqrt{\frac{2b}{3p}}\right)+\min\left(1,3\sqrt{\frac{3bp}{8}}\right)(1+32p^2)T_0}$$

$$=\frac{1}{RTT\left(\sqrt{\frac{2bp}{3}}\right)+\min\left(1,3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)T_0}$$

<-Eq. (29) most well-known version of throughput formula

Impact of receiver's rwnd limitation



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

Full model Eq. (31)

$$B(p) \simeq \frac{\frac{1-p}{p} + E[W] + \hat{Q}(E[W])\frac{1}{1-p}}{RTT(E[X]+1) + \hat{Q}(E[W])T_O \frac{f(p)}{1-p}}$$

if $E[W] < W_{max}$,

$$B(p) \simeq \frac{\frac{1-p}{p} + W_{max} + \hat{Q}(W_{max})\frac{1}{1-p}}{RTT(\frac{b}{8}W_{max} + \frac{1-p}{pW_{max}} + 2) + \hat{Q}(W_{max})T_O \frac{f(p)}{1-p}}$$

otherwise, use W_{max} for E[W] and recompute E[X]
(derivation omitted)

Impact of receiver's rwnd limitation—approximate model

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

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

which is referred to as Eq. (32)

Summary data from traces (1 hour)

Sender	Receiver	Packets Sent	Loss Indic.	TD	T_0	T_1	T_2	T_3	T_4	T_5 or more	RTT	Time Out
manic	alps	54402	722	19	611	67	15	6	2	2	0.207	2.505
manic	baskerville	58120	735	306	411	17	1	0	0	0	0.243	2.495
manic	ganef	58924	743	272	444	22	4	1	0	0	0.226	2.405
manic	mafalda	56283	494	2	474	17	1	0	0	0	0.233	2.146
manic	maria	68752	649	1	604	35	8	1	0	0	0.180	2.416
manic	spiff	117992	784	47	702	34	1	0	0	0	0.211	2.274
manic	sutton	81123	1638	988	597	41	7	3	1	1	0.204	2.459
manic	tove	7938	264	1	190	37	18	8	3	7	0.275	3.597
void	alps	37137	838	7	588	164	56	17	4	2	0.162	0.489
void	baskerville	32042	853	339	430	67	12	5	0	0	0.482	1.094
void	ganef	60770	1112	414	582	79	20	9	4	2	0.254	0.637
void	maria	93005	1651	33	1344	197	54	15	5	3	0.152	0.417
void	spiff	65536	671	72	539	56	4	0	0	0	0.415	0.749
void	sutton	78246	1928	840	863	152	45	18	9	1	0.211	0.601
void	tove	8265	856	5	444	209	100	51	27	12	0.272	1.356
babel	alps	13460	1466	0	1068	247	87	33	18	8	0.194	1.359
babel	baskerville	62237	1753	197	1467	76	10	3	0	0	0.253	0.429
babel	ganef	86675	2125	398	1686	38	2	1	0	0	0.201	0.306
babel	spiff	57687	1120	0	939	137	36	7	1	0	0.331	0.953
babel	sutton	83486	2320	685	1448	142	31	9	4	1	0.210	0.705
babel	tove	83944	1516	1	1364	118	17	7	5	3	0.194	0.520
pif	alps	83971	762	0	577	111	46	16	8	2	0.168	7.278
pif	imagine	44891	1346	15	1044	186	63	21	10	5	0.229	0.700
pif	manic	34251	1422	43	944	272	105	36	14	6	0.257	1.454

Table 2: Summary data from 1hr traces

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

Summary data from 100s traces

Sender	Receiver	Packets Sent	Loss Indic.	TD	T_0	T_1	T_2	T_3	T_4	T_5 or larger	RTT	Time Out
manic	ada	531533	6432	4320	2010	93	7	2	0	0	0.1419	2.2231
manic	afer	255674	4577	2584	1898	83	10	1	1	0	0.1804	2.3009
manic	al	264002	4720	2841	1804	70	5	0	0	0	0.1885	2.3542
manic	alps	667296	3797	841	2866	85	5	0	0	0	0.1125	1.9151
manic	baskerville	89244	1638	627	955	42	11	2	1	0	0.4735	3.2269
manic	ganef	160152	2470	1048	1308	89	18	6	1	0	0.2150	2.6078
manic	mafalda	171308	1332	9	1269	48	5	1	0	0	0.2501	2.5127
manic	maria	316498	2476	5	2362	99	8	2	0	0	0.1166	1.8798
manic	modi4	282547	6072	3976	1988	99	8	1	0	0	0.1749	2.2604
manic	pong	358535	4239	2328	1830	74	7	0	0	0	0.1769	2.1371
manic	spiff	298465	2035	159	1781	75	14	4	2	0	0.2539	2.4545
manic	sutton	348926	6024	3694	2238	87	5	0	0	0	0.1683	2.1852
manic	tove	262365	2603	6	2422	135	30	8	2	0	0.1153	1.9551

- 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_O are average values over 100 traces for same S-D pair

Experimental comparison (1)

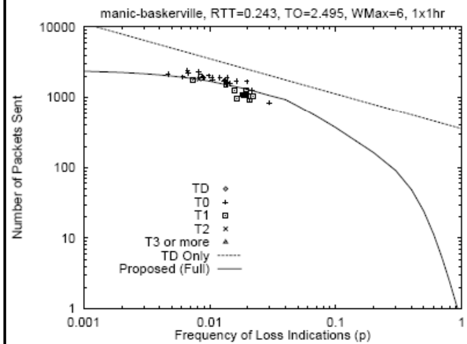


Figure 7: manic to baskerville

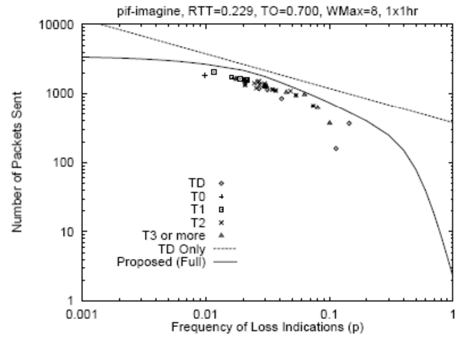


Figure 8: pif to imagine

- 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)

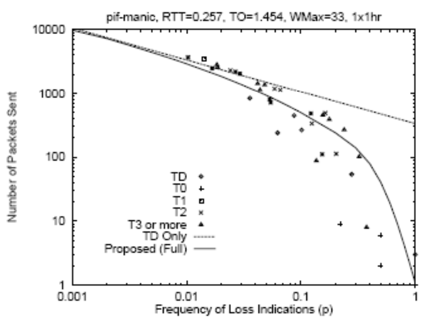


Figure 9: pif to manic

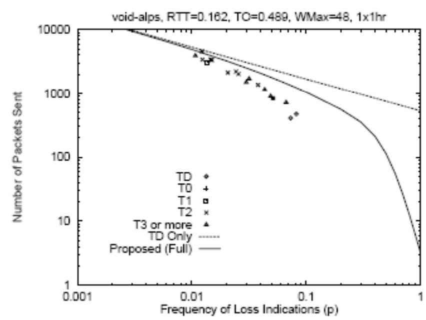


Figure 10: void to alps

W_{max} = 33

W_{max}=44

Experimental comparison (3)

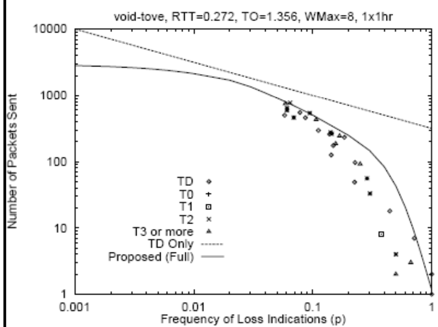


Figure 11: void to tove

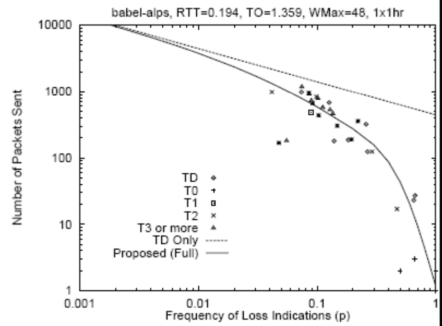


Figure 12: babel to alps

W_{max}=8

W_{max}=48

Accuracy of approximate model

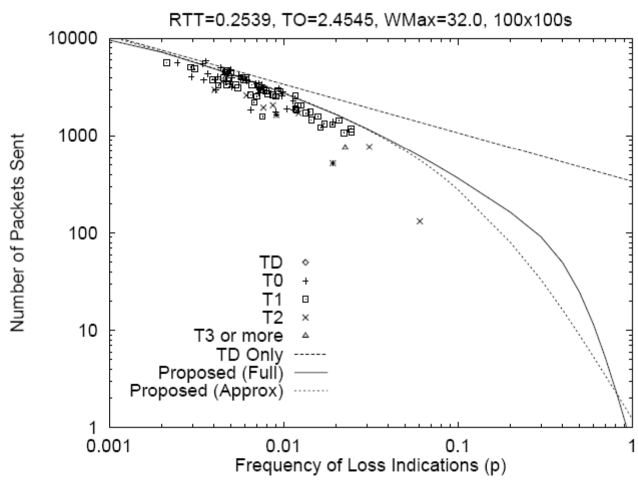


Figure 18: manic to spiff, with predictions by both full and approximate models (W_{max}=32)

Average errors

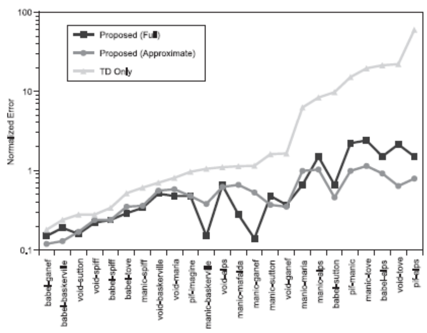


Figure 19: Comparison of the models for 1hr traces

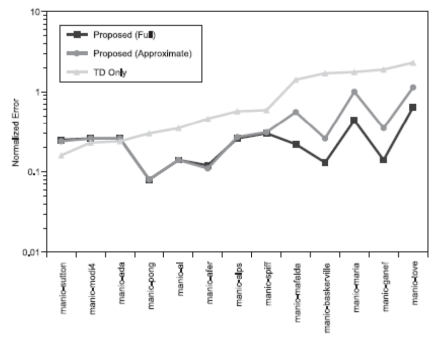


Figure 20: Comparison of the models for 100s traces

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

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