Modeling Congestion Control: Part 1

Lecture 2

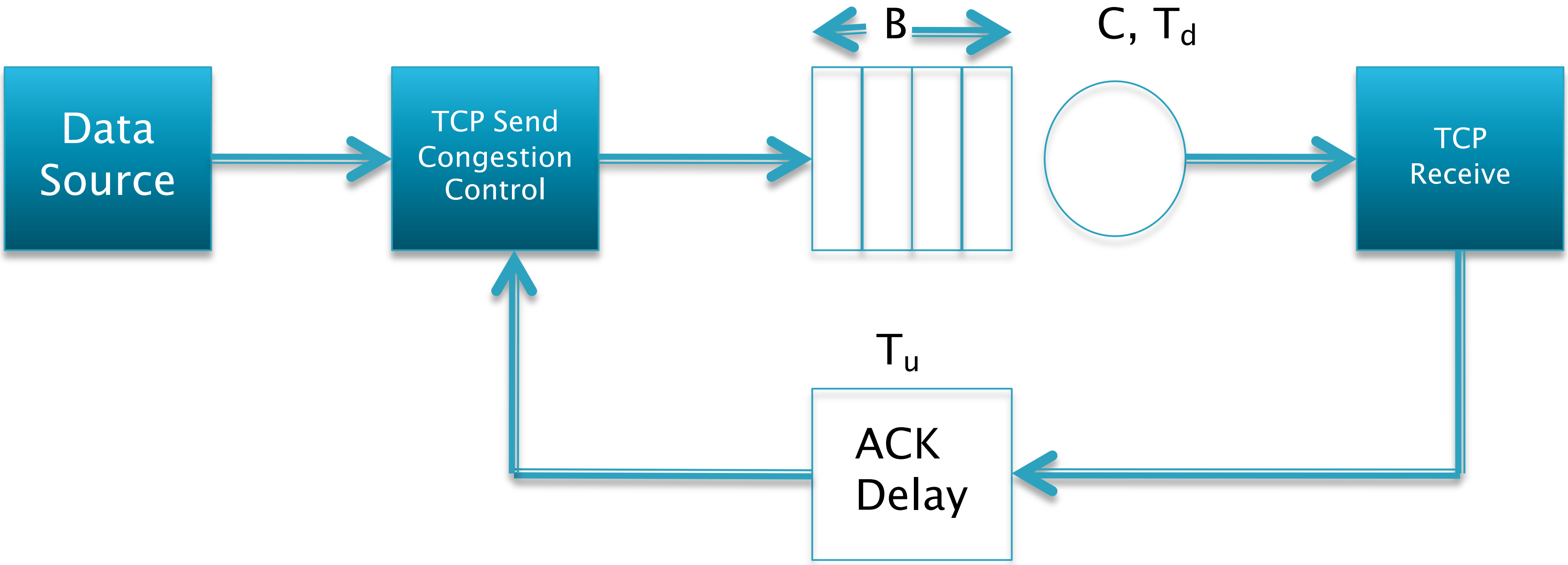
Subir Varma

Why do Modeling?

The focus on a large part of this course is on models for Congestion Control. Hence rather than just describing the algorithm, we also try to analyze its behavior using models. Why do this?

- Models enable us to capture the fundamental factors that influence the algorithm’s behavior, and in some cases, summarize it in the form of an equation. This provides valuable guidelines on ways the congestion control algorithm can be improved or adapted to different network conditions.
- The analysis of congestion control algorithms provides fundamental insight into their ultimate performance limits.
  - Modeling of TCP led to the realization that it becomes unstable if the link speed or the round trip delay becomes very large.
  - More recently, modeling of delay based congestion control has shown that their performance degrades if the network jitter is high.

Single TCP Connection



W(t): TCP window size in packets at time t

W_{max}: Maximum TCP window size

W_f: Window size at which a packet is lost

MSS: TCP packet size in bytes

C: The bottleneck node capacity in packets/sec

B: Buffer count at the bottleneck node in packets

T_d: Propagation delay in the forward (or downstream) direction in seconds

T_u: Propagation delay in the backward (or upstream) direction in seconds

T: Total propagation + Transmission delay for a segment

b_{max}: The maximum buffer occupancy at the bottleneck node in packets

R(t): Throughput of the connection in packets/sec at time t

R_{avg}: Average throughput of a connection in packets/sec

Total Fixed Round Trip Latency

$$T = T_d + T_u + \frac{1}{C}$$

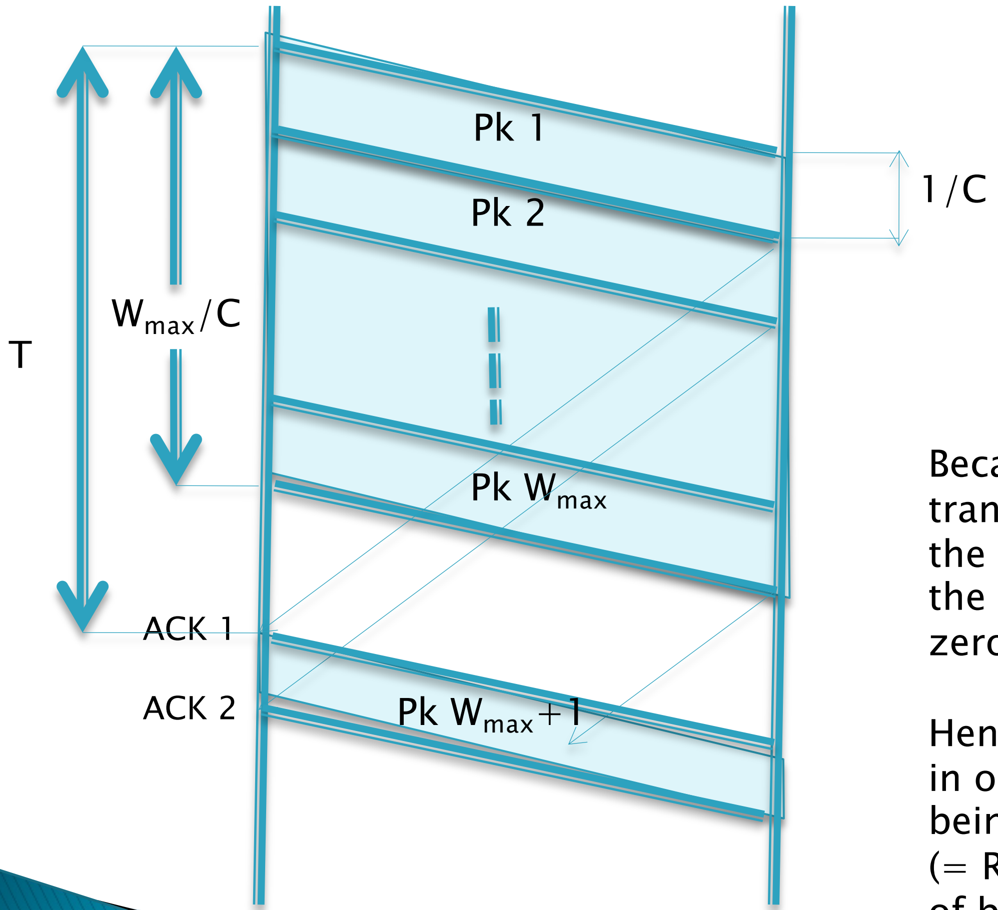
Models with No Packet Loss

Single TCP Connection with No Packet Drops

Assumptions:

- The window size is in equilibrium with size W_{max}
- There is sufficient buffering at the bottleneck node, so no packets get dropped because of overflows, and
- The link is error free.

Packet Transmissions with $T > W_{max}/C$



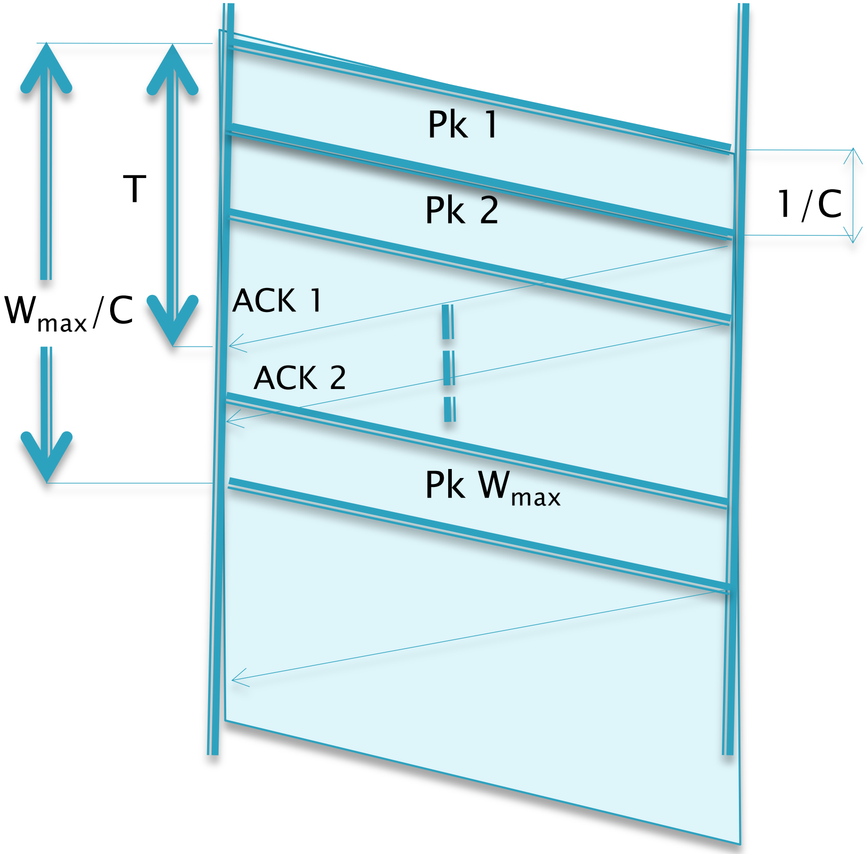
The sender transmits the entire window full of packets before it receives the ACK for first packet back.
Hence, the average connection throughput R_{avg} is limited by the window size and is given by

$$R_{avg} = \frac{W_{max}}{T} \quad \text{when } W_{max} < CT$$

Because the rate at which the bottleneck node is transmitting packets is greater than or equal to the rate at which the source is sending packets into the network, the bottleneck node queue size is zero (also due to ACK self clocking).

Hence, at any instant in time, all the packets are in one of two places: either in the process of being transmitted in the downstream direction ($= R_{avg}.T_d$ packets), or their ACKs are in the process of being transmitted in the upstream direction ($= R_{avg}.T_u$ ACKs).

Packet Transmissions with $T < W_{max}/C$



The ACKs for the first packet in the window arrive at the sender before it has finished transmitting the last packet in that window.

As a result, the sender is continuously transmitting. The system throughput is no longer limited by the window size, and is able to attain the full value of the link capacity

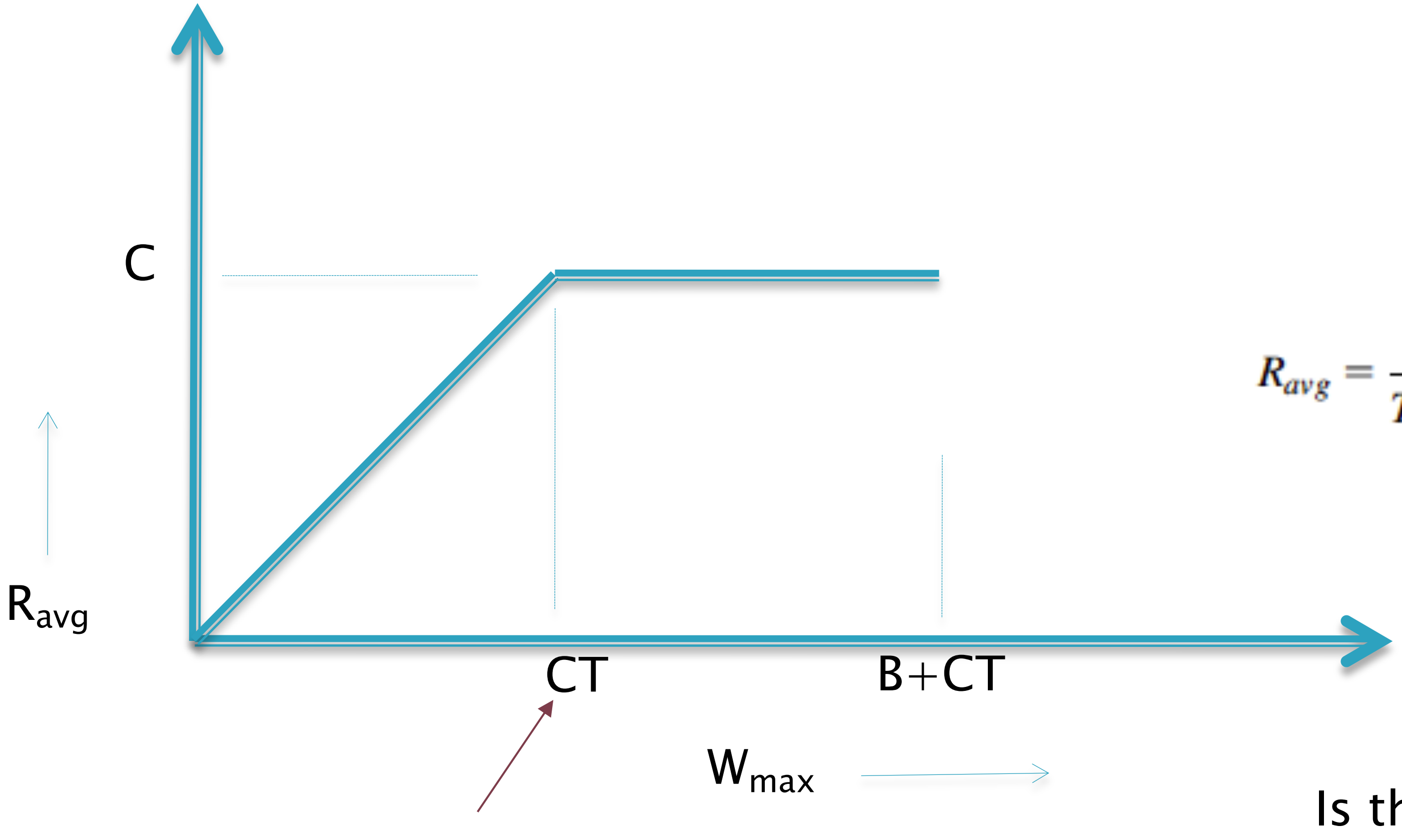
$$R_{avg} = C \quad \text{when } W_{max} > CT$$

In this case, $C.T_d$ packets are the process of being transmitted in the downstream direction, and $C.T_u$ ACKs are in the process of being transmitted in the upstream direction, for a total of $C. (T_d+T_u)=CT$ packets and ACKs.

Because $CT<W_{max}$,
this leaves (W_{max}–CT) packets from the TCP window unaccounted for, and indeed they are to be found queued up in the bottleneck node! Hence, in equilibrium, the maximum buffer occupancy at the bottleneck node is given by

$$b_{max} = \max(W_{max} - CT, 0)$$

Single TCP Connection with No Packet Drops



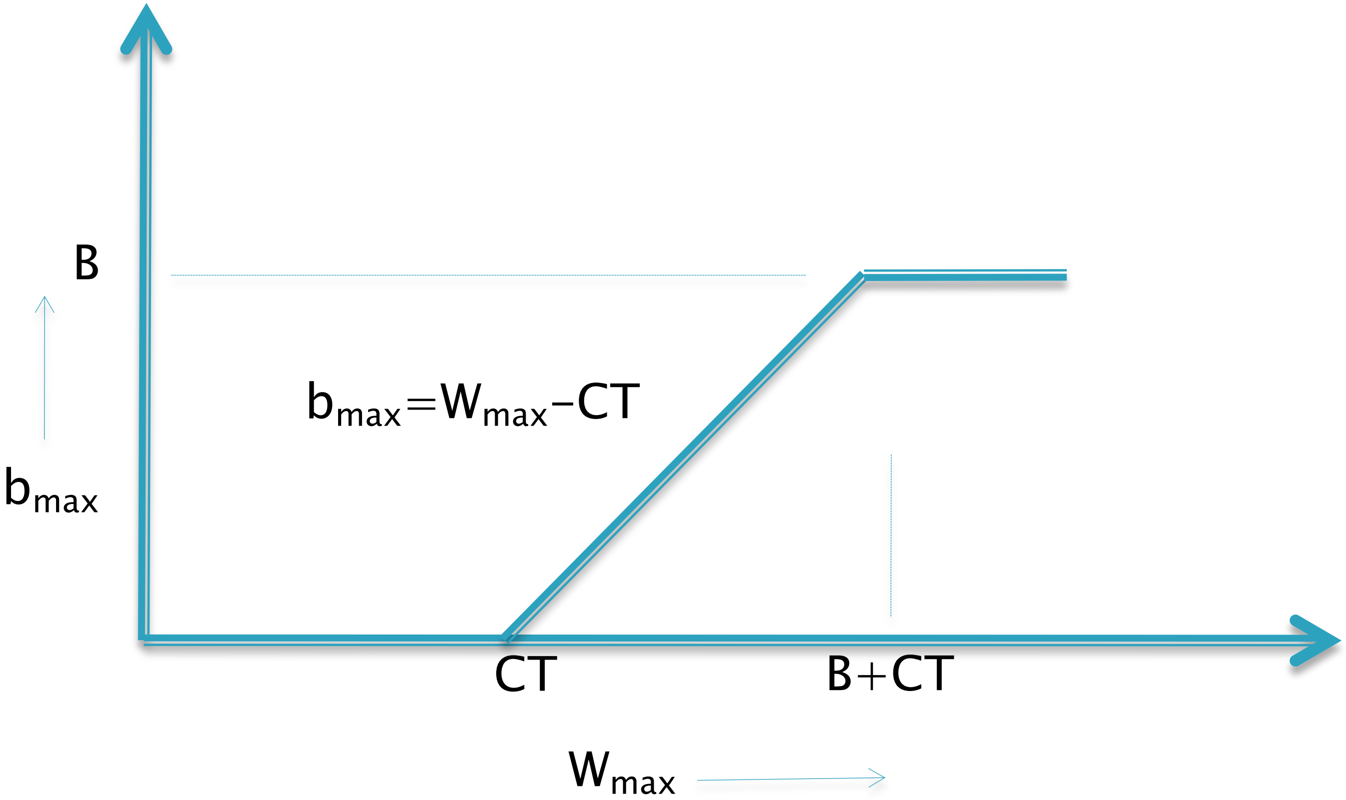
$$R_{avg} = \frac{W_{\max}}{T + \frac{b_{\max}}{C}}$$

Is that bad?

Ideal Window Size

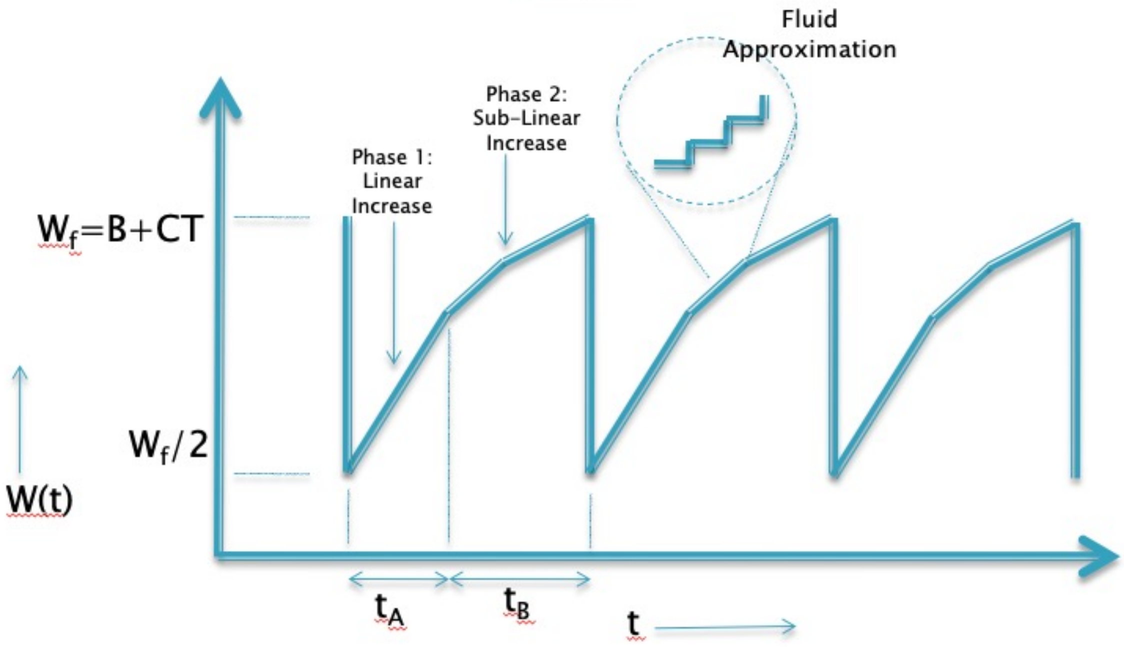
Larger Window Size causes queueing delays without adding to the throughput

Maximum Buffer Occupancy



When W_{max} increases to B+CT, then $b_{max}=B$, and any increase of W_{max} beyond this value will cause the buffer to overflow.

TCP Reno ($W_{max} > B + CT$)



W_f: Window size at which packet is lost

Drop in Window Size leaves the buffer full

Assumptions:

- We assume that we are in the steady phase of a long file transfer, during which the system spends all its time in the congestion avoidance phase, i.e., the initial Slow-Start phase is ignored, and all packet losses are recovered using duplicates ACKs (i.e., without using timeouts).
- We also ignore the Fast Recovery phase because it has a negligible effect on the overall throughput, and including it considerably complicates the analysis.
- Even though the window size W increases and decreases in discrete quantities, we replace it by a continuous variable W(t), thus resulting in what is known as a fluid approximation.

Some Definitions

Define b(t) as the fluid approximation to the buffer occupancy process at time t, given by

$$b(t) = W(t) - CT$$

Define T(t) is the total round trip latency at time t (includes propagation + transmission + queueing delays), given by

$$T(t) = T + \frac{b(t)}{C}$$

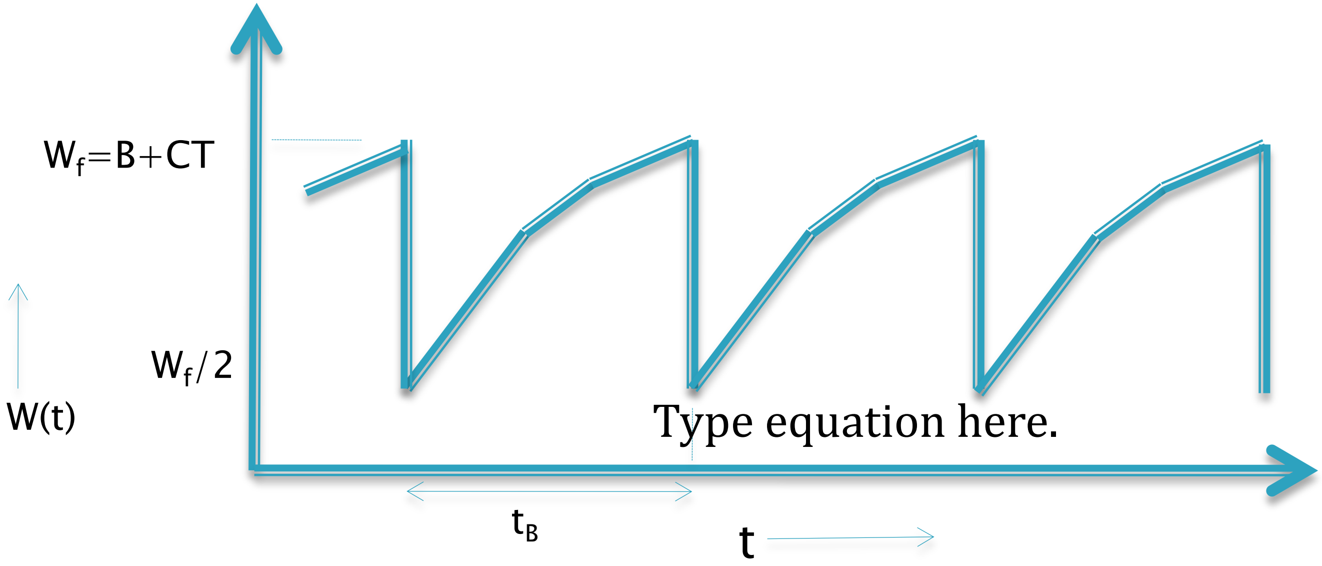
The throughput R(t) at time t is defined by

$$R(t) = \frac{W(t)}{T(t)}$$

and the average throughput is given by

$$R_{avg} = \lim_{t \to \infty} \frac{1}{t} \int_0^t R(t) dt$$

Window Evolution for W_{max} > B + CT and $W_f/2$ > CT

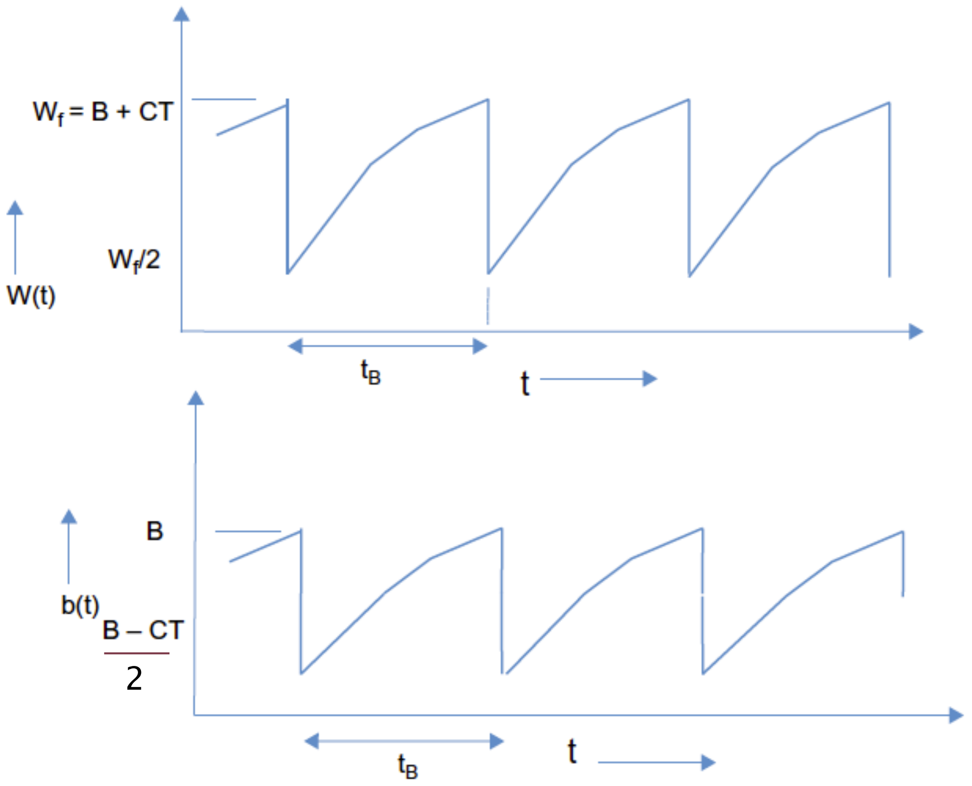


- W(t) increases up to a maximum value of $W_f = B + CT$ and is accompanied by an increase in buffer size to B, at which point a packet is lost at the bottleneck node because of buffer overflow.
- TCP Reno then goes into the congestion recovery mode, during which it retransmits the missing packet and then reduces the window size to $W_f/2$.
- if $W_f/2$ exceeds the delay-bandwidth product for the connection, then $R_{avg} = C$.

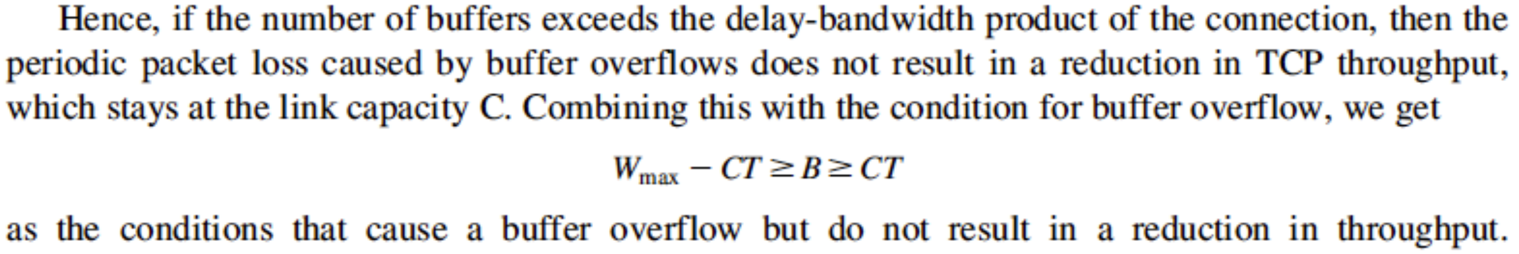
$\frac{W_f}{2} \geq CT$ implies that $\frac{B+CT}{2} \geq CT$, i.e., $B \geq CT$.

Minimum number of buffers required to avoid underflow

Buffer Occupancy Evolution for $W_{max} > B + CT$ and $B > CT$



b(t)=W(t)−C

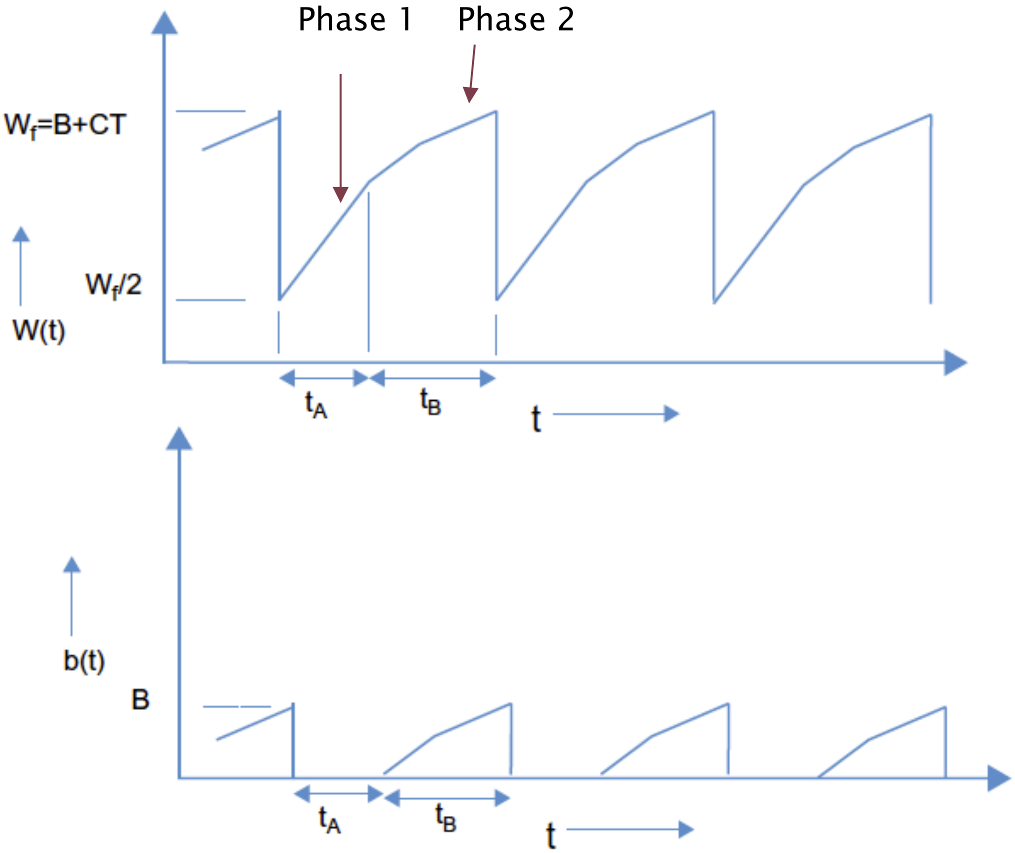


Hence, if the number of buffers exceeds the delay-bandwidth product of the connection, then the periodic packet loss caused by buffer overflows does not result in a reduction in TCP throughput, which stays at the link capacity C. Combining this with the condition for buffer overflow, we get

$$W_{\max} - CT \geq B \geq CT$$

as the conditions that cause a buffer overflow but do not result in a reduction in throughput.

Window Evolution for $W_{max} > B + CT$ and $W_f/2 < CT$



Two phase evolution in Window Size

B < CT: This causes the queue to periodically empty out

Computing t_A and n_A in Phase 1

- In phase 1, W(t) < CT, which implies that the queue at the bottleneck node is empty.
- Hence, the rate at which ACKs are returning to the sender is equal to the TCP throughput R(t) = W(t)/T. This implies that the rate at which the window is increasing is given by

$$\frac{dW(t)}{dt} = \frac{dW}{da} \cdot \frac{da}{dt} = \frac{1}{W(t)} \cdot \frac{W(t)}{T} = \frac{1}{T} \qquad \text{so that} \qquad W(t) - \frac{W_f}{2} = \int_0^t \frac{dt}{T} = \frac{t}{T}$$

This results in a linear increase in the window size during phase 1

Phase 1 ends when $W = W_f = CT$, so if t_A is the duration of this phase, then

$$CT - \frac{W_f}{2} = \frac{t_A}{T}, \qquad \text{so that} \qquad t_A = T\left(CT - \frac{W_f}{2}\right).$$

The number of packets n_A transmitted during phase 1 is given by

$$n_A = \int_0^{t_A} R(t)dt = \int_0^{t_A} \frac{W(t)}{T} dt = \frac{1}{T}\left(\frac{W_f \cdot t_A}{2} + \frac{t_A^2}{2T}\right)$$

Computing t_B and n_B in Phase 2

- In phase 2, W(t).CT so that the bottleneck node has a persistent backlog.
- Hence, the rate a which ACKs are returning back to the sender is given by C. It follows that the rate of increase of the window is given by

$$\frac{dW(t)}{dt} = \frac{dW(t)}{da} \cdot \frac{da}{dt} = \frac{C}{W(t)}$$

Sub-linear increase

So that $W^2(t) - (CT)^2 = 2Ct$ i.e., $W(t) = \sqrt{2Ct + (CT)^2}$

It follows that $t_B = \dfrac{W_f^2 - (CT)^2}{2C}$ and $n_B = Ct_B = \dfrac{W_f^2 - (CT)^2}{2}$

Average TCP Throughput

The average TCP throughput is given by

$$R_{avg} = \frac{n_A + n_B}{t_A + t_B}$$

$$n_A + n_B = \frac{1}{T}\left(\frac{W_f \cdot t_A}{2} + \frac{t_A^2}{2T}\right) + \frac{W_f^2 - (CT)^2}{2} = \frac{3W_f^2}{8}$$

Number of packets transmitted during a cycle is proportional to the square of the max Window size W_f

and

$$t_A + t_B = T\left(CT - \frac{W_f}{2}\right) + \frac{W_f^2 - (CT)^2}{2C} = \frac{W_f^2 + (CT)^2 - CTW_f}{2C}$$

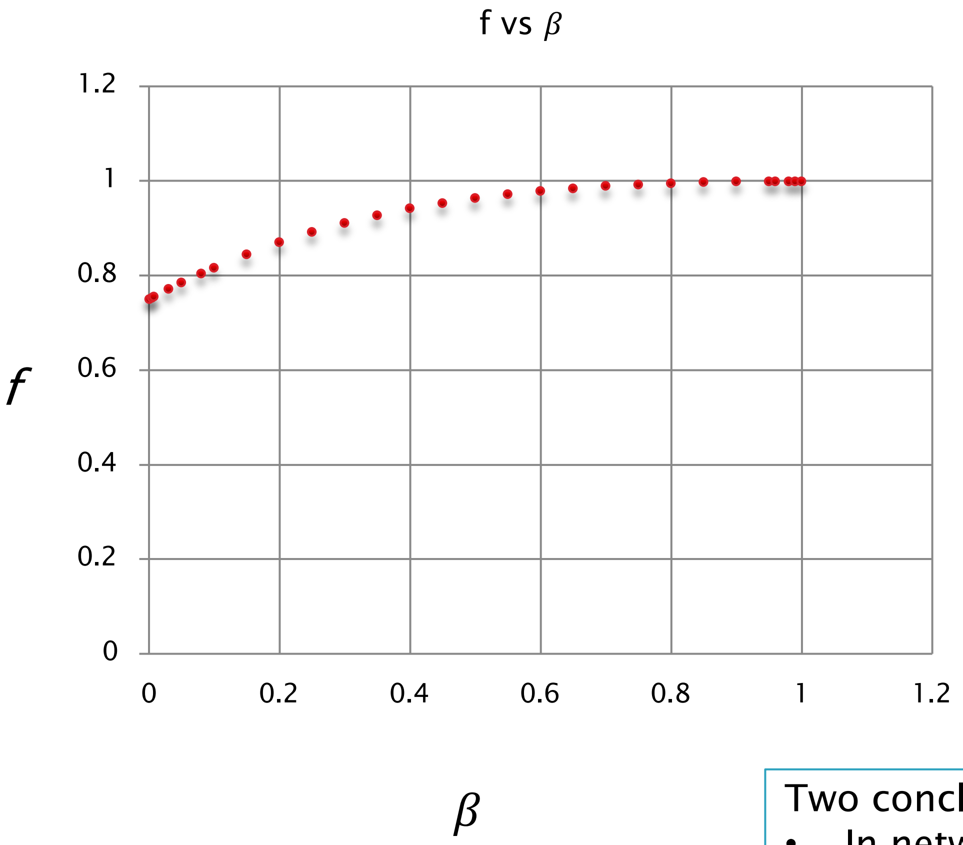
So that

$$R_{avg} = \frac{\frac{3}{4}W_f^2 C}{W_f^2 + (CT)^2 - CTW_f}$$

Substituting W_f = B+CT

$$R_{avg} = \frac{0.75}{\left[1 - \frac{B \cdot CT}{(B+CT)^2}\right]}C$$

Average TCP Throughput



Define

$$\beta = \frac{B}{CT} \leq 1,$$

Then

$$R_{avg} = \frac{0.75}{\left[1 - \frac{\beta}{(1+\beta)^2}\right]} C$$

Define

$$f = \frac{0.75}{\left[1 - \frac{\beta}{(1+\beta)^2}\right]}$$

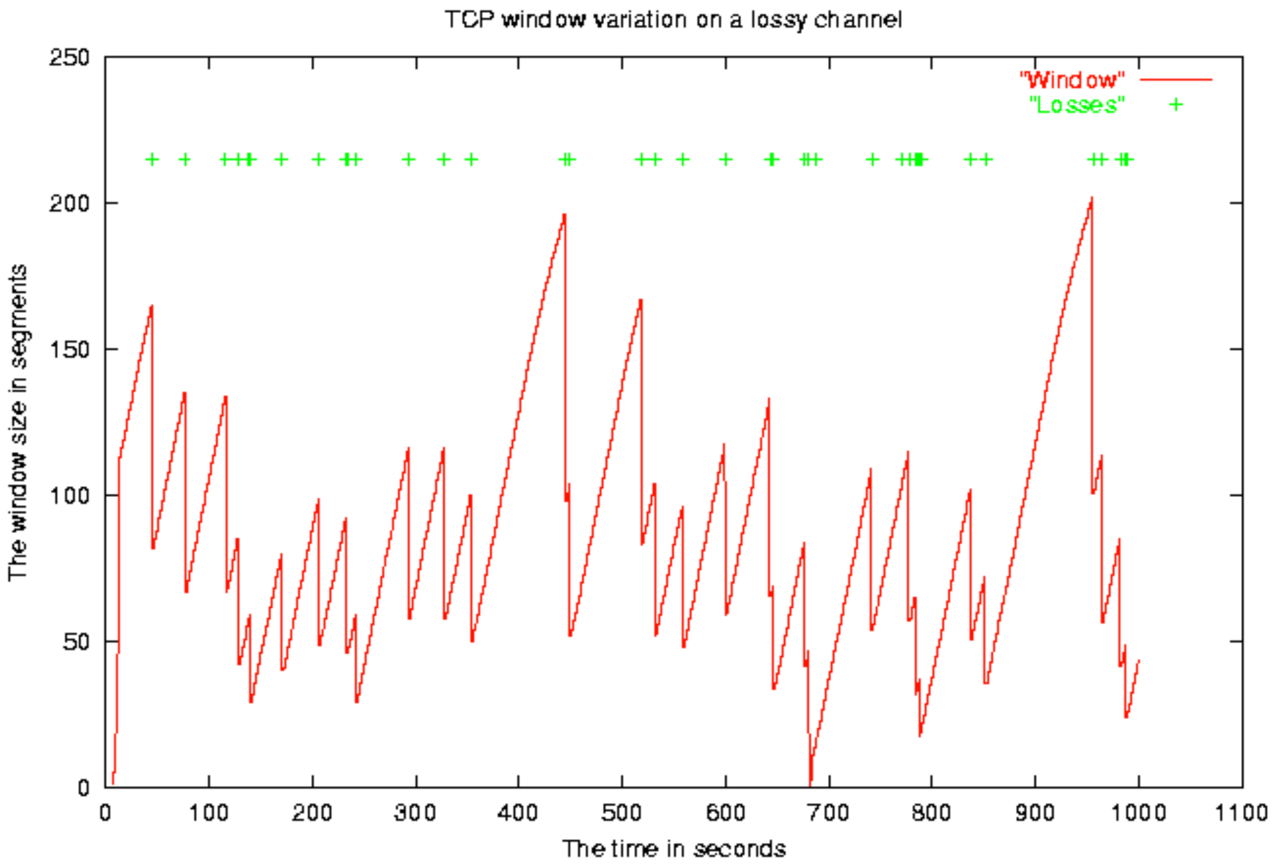
$$\lim_{\beta \to 0} f = 0.75 \quad \text{and} \quad \lim_{\beta \to 1} f = 1$$

Two conclusions:

- In networks with large delay bandwidth product, TCP may not be able to achieve full link capacity due to lack of buffers
- Even with very small buffer sizes, TCP is still able to get to 0.75C

Models with Packet Loss

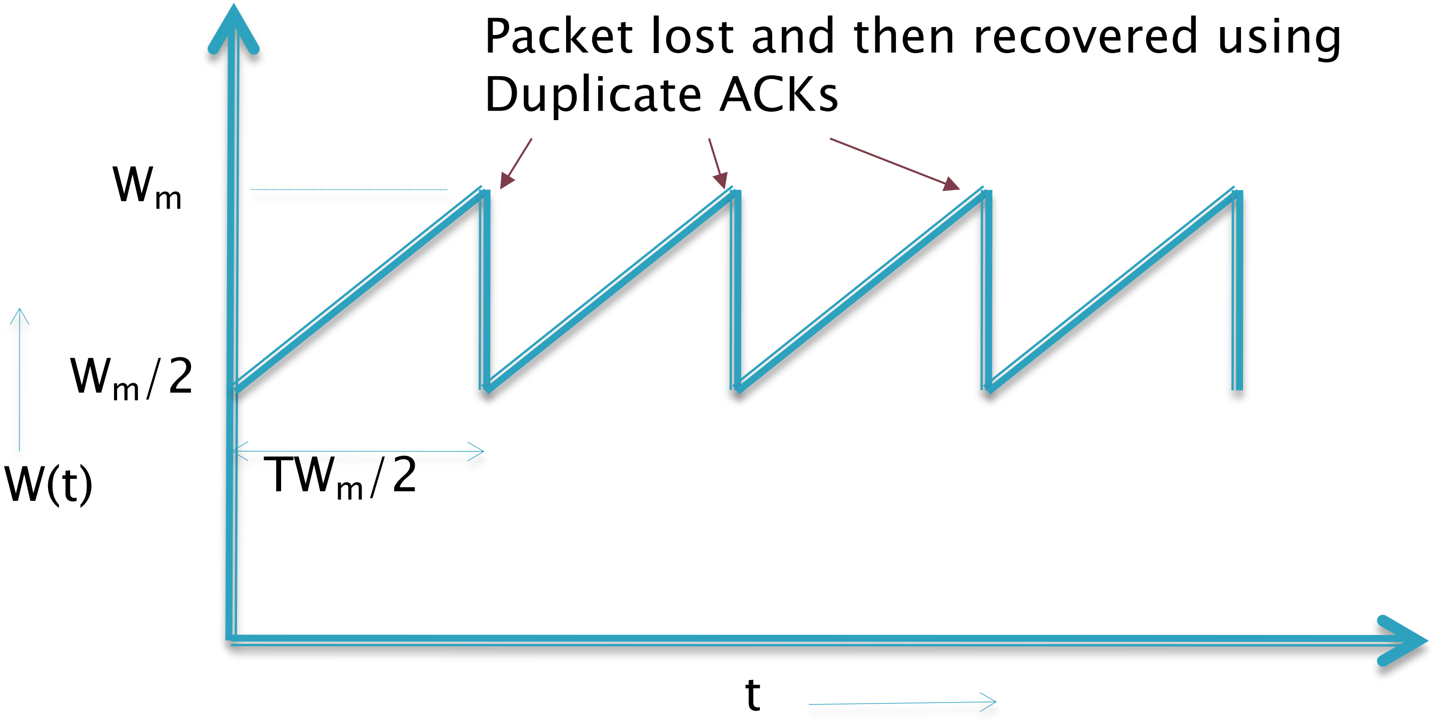
TCP Window Variation on a Lossy Channel



Analysis of Lossy TCP

- Fluid Flow Models
- Mean Value Analysis (MVA)
- Markov Chain Models
- Stochastic Models

Fluid Flow Model



W_m is now a Random Variable

Fluid Flow Model

- The value of the maximum window size is W_m, so that the value of the window after a packet loss is $W_m/2$.
- During the period in which the window size increases from $W_m/2$ to W_m (called a cycle), it increases by 1 for every round trip duration T; hence, it takes $W_m/2$ round trips to increase from $W_m/2$ to W_m.
- This implies that the length of a cycle is given by $TW_m/2$ seconds.

The number of packets that are transmitted during a cycle is given by

$$Y = \int_0^{TW_m/2} R(t)dt$$

$$\begin{aligned} Y &= \int_0^{TW_m/2} \frac{W(t)}{T} dt \\ &= \frac{1}{T}\left[T\left(\frac{W_m}{2}\right)^2 + \frac{T}{2}\left(\frac{W_m}{2}\right)^2\right] \\ &= \frac{3}{8}W_m^2 \end{aligned}$$

Fluid Flow Model: The Square Root Formula

Throughput R for a single cycle is given by

$$R = \frac{Y}{TW_m/2} = \frac{3}{4}\frac{W_m}{T} = \frac{3}{4T}\sqrt{\frac{8Y}{3}} = \frac{1}{T}\sqrt{\frac{3Y}{2}}$$

So that

$$R_{avg} = \frac{1}{T}\sqrt{\frac{3}{2}}E\left(\sqrt{Y}\right)$$

What is the distribution of Y?

True only if Y is a constant

Simplifying assumption $E\left(\sqrt{Y}\right) = \sqrt{E(Y)}$

Assuming iid packet drop probability $P(Y = n) = (1-p)^{n-1}p$

So that E(Y) = 1/p

$$\boxed{R_{avg} = \frac{1}{T}\sqrt{\frac{3}{2p}}}$$

Fluid Flow Model: The Square Root Formula

Without simplifying assumption

$$R_{avg} = \frac{1}{T}\sqrt{\frac{3\pi}{8p}}$$

The most important information in the formula for the average throughput is not the constant but the nature of the functional dependence on p and T.

The deterministic approximation gives the correct functional dependence while using fairly straightforward computations

Taking TCP Timeouts into Account

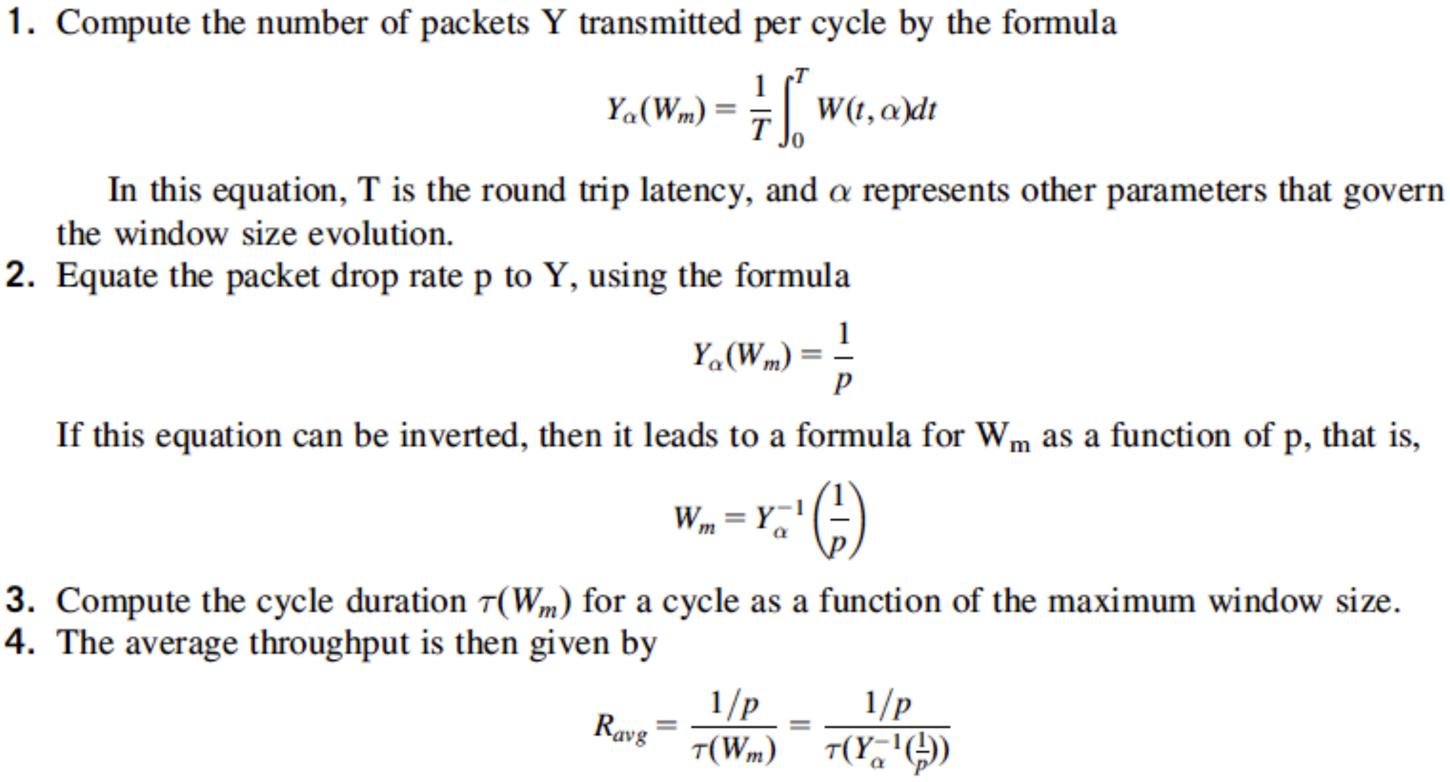
$$R_{avg} = \frac{1 - p + p\min(1,3\sqrt{\frac{3p}{8}})}{T(1-p)\sqrt{\frac{2p}{3}} + T_0 p f(p)\min(1,3\sqrt{\frac{3p}{8}})}$$

$$f(p) = 1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6$$

T_0: Timeout interval

$$R_{avg} = \frac{1}{T\sqrt{\frac{2p}{3}} + T_0 p(1+32p^2)\min(1,3\sqrt{\frac{3p}{8}})}, \qquad \text{if p} \ll 1$$

General Procedure for Computing Average Throughput



1. Compute the number of packets Y transmitted per cycle by the formula

$$Y_\alpha(W_m) = \frac{1}{T}\int_0^T W(t,\alpha)dt$$

In this equation, T is the round trip latency, and α represents other parameters that govern the window size evolution.

2. Equate the packet drop rate p to Y, using the formula

$$Y_\alpha(W_m) = \frac{1}{p}$$

If this equation can be inverted, then it leads to a formula for W_m as a function of p, that is,

$$W_m = Y_\alpha^{-1}\left(\frac{1}{p}\right)$$

3. Compute the cycle duration $\tau(W_m)$ for a cycle as a function of the maximum window size.
4. The average throughput is then given by

$$R_{avg} = \frac{1/p}{\tau(W_m)} = \frac{1/p}{\tau(Y_\alpha^{-1}(\frac{1}{p}))}$$

Beyond Reno: AIMD Congestion Control

$$W \leftarrow \begin{cases} W + a & per \quad RTT \quad on \quad ACK \quad receipt \\ (1-b)W & \qquad\qquad\quad on \quad packet \quad loss \end{cases}$$

(a,b) are constants

Some of the most effective algorithms combine AIMD with increase decrease parameters (a,b) that are allowed to vary as a function of either the congestion feedback or the window size itself.

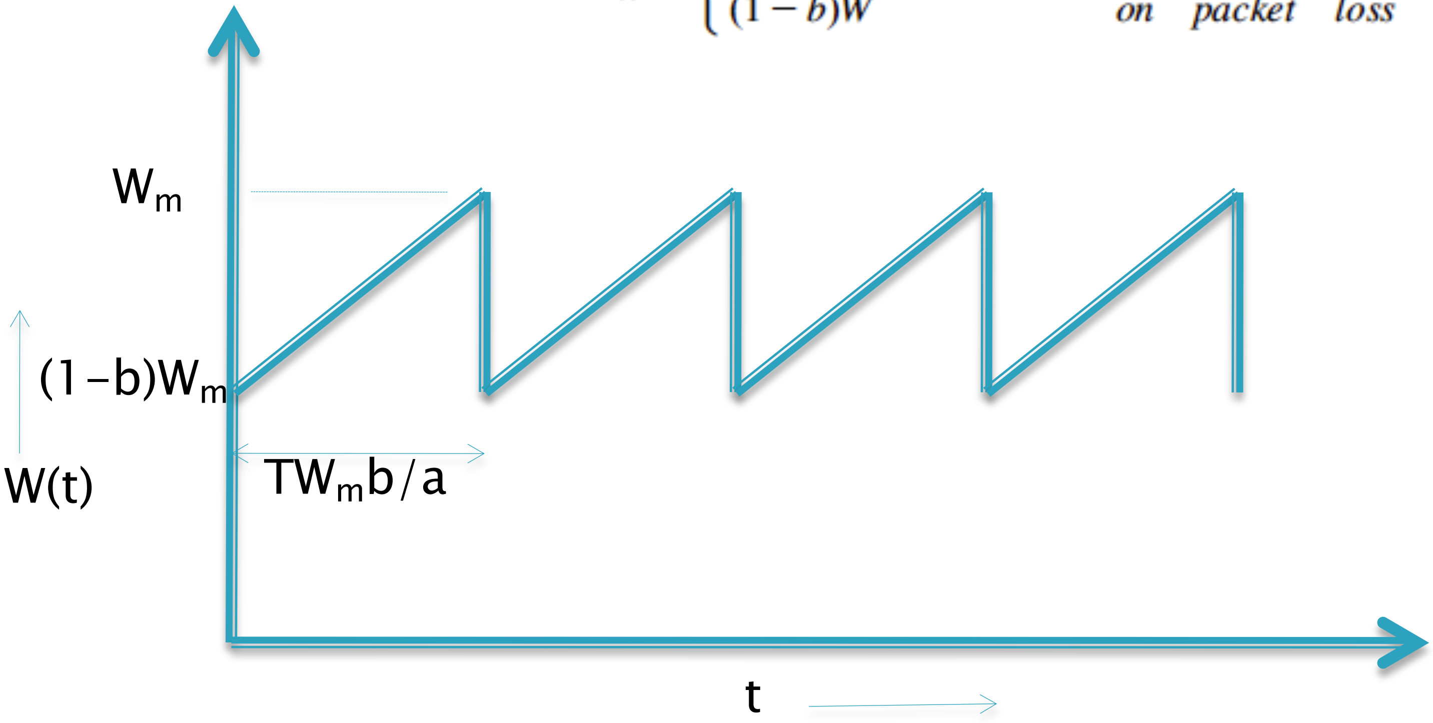
AIMD Congestion Control

- Choosing the value of a,
  - the designer can make the congestion control more aggressive (if a > 1) or less aggressive than TCP Reno (if a <1) than less aggressive than TCP Reno
  - The choice a >1 is used in high-speed. networks because it causes the window to increase faster to take advantage of the available bandwidth.
  - Protocols such as High Speed TCP (HSTCP), TCP BIC, TCP FAST and TCP LEDBAT use an adaptive scheme in which they vary the value of a depending on the current window size, the congestion feedback, or both.

- The value of b
  - B influences how readily the connection gives up its bandwidth during congestion.
  - TCP Reno uses a rather aggressive reduction of window size by half, which causes problems in higher speed links.
  - By choosing b<0.5, the reduction in window size is smaller on each loss, which leads to a more stable congestion control algorithm. But this can lead to fairness issues because existing connections don’t give up their bandwidth as readily.
  - Protocols such as Westwood, Data Center TCP (DCTCP) and HSTCP vary b as a function of the congestion feedback (Westwood, DCTCP) or the window size (HSTCP).

AIMD Congestion Control

$$W \leftarrow \begin{cases} W + a & \text{per RTT on ACK receipt} \\ (1-b)W & \text{on packet loss} \end{cases}$$



(a,b) are constants

AIMD Congestion Control Analysis

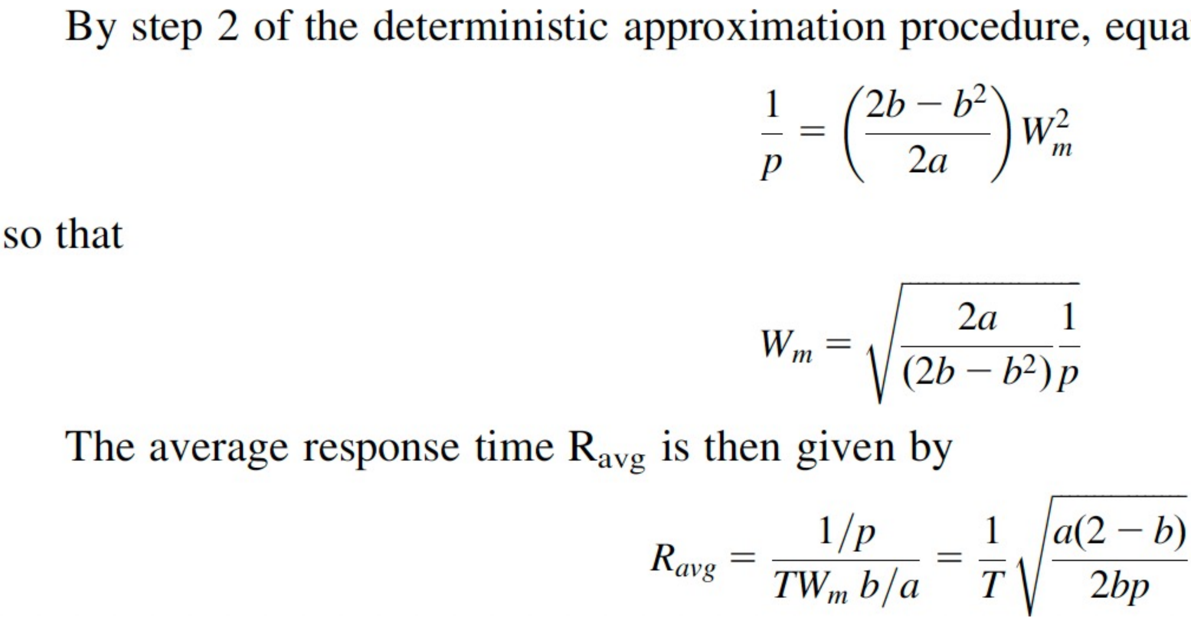
- The window size reduces from W$_m$ to (1−b)W$_m$ on packet loss (i.e., a reduction of bW$_m$ packets).
- On each round trip, the window size increases by a packets, so that it takes bW$_m$/a round trips for the window to get back to W$_m$. So that the cycle length τ is given by

$$\tau = T\frac{bW_m}{a}$$

The number of packets transmitted over a single cycle Y is given by

$$\begin{aligned} Y &= \int_0^{TW_mb/a} \frac{W(t)}{T}dt \\ &= \frac{1}{T}\left[T\frac{W_m\,b}{a}(1-b)W_m + \frac{T}{2}\frac{W_m}{2}\frac{b}{a}W_mb\right] \\ &= \left(\frac{2b-b^2}{2a}\right)W_m^2 \end{aligned}$$

AIMD Analysis



By step 2 of the deterministic approximation procedure, equating Y to 1/p leads to

$$\frac{1}{p} = \left(\frac{2b - b^2}{2a}\right) W_m^2$$

so that

$$W_m = \sqrt{\frac{2a}{(2b - b^2)} \frac{1}{p}}$$

The average response time R_{avg} is then given by

$$R_{avg} = \frac{1/p}{TW_m\, b/a} = \frac{1}{T}\sqrt{\frac{a(2-b)}{2bp}}$$

TCP Reno

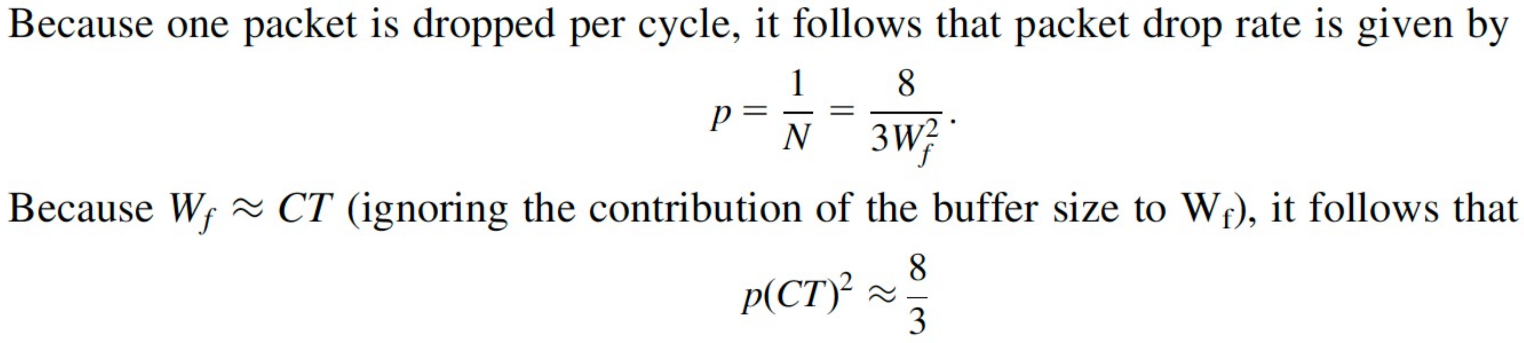
$$\boxed{R_{avg} = \frac{1}{T}\sqrt{\frac{3}{2p}}}$$

Differentiating Buffer Overflows from Link Errors

If

$$p(CT^2) \approx \frac{8}{3}$$

Then buffer overflows account for most packet drops



Because one packet is dropped per cycle, it follows that packet drop rate is given by

$$p = \frac{1}{N} = \frac{8}{3W_f^2}.$$

Because $W_f \approx CT$ (ignoring the contribution of the buffer size to $\mathrm{W_f}$), it follows that

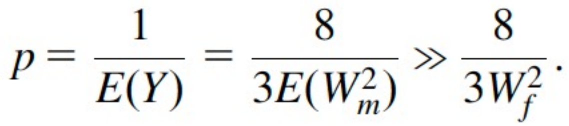
$$p(CT)^2 \approx \frac{8}{3}$$

Differentiating Buffer Overflows from Link Errors

If

$$p(CT^2) \gg \frac{8}{3}$$

Then link errors account for most packet drops



$$p = \frac{1}{E(Y)} = \frac{8}{3E(W_m^2)} \gg \frac{8}{3W_f^2}.$$

Since
$W_m << W_f$

It follows that if

$$p(CT)^2 \gg \frac{8}{3}$$

What if Packet Losses are Correlated?

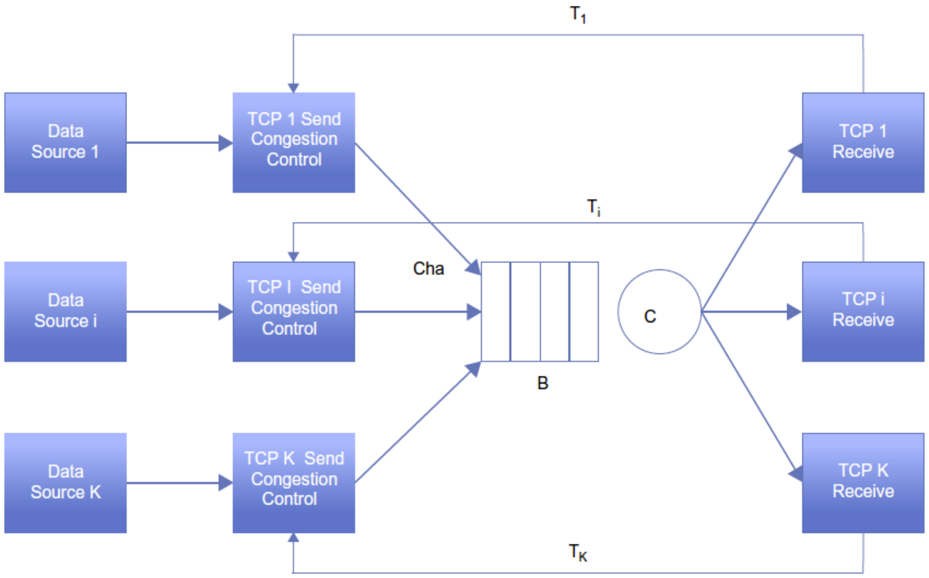
$$R_{avg} = \frac{1}{T}\sqrt{\frac{1}{p}\left[\frac{\hat{U}(0)}{2} + \sum_{k=1}^{\infty}(1-b)^k \hat{U}(k)\right]}$$

$S_n = \tau_{n+1} - \tau_n$: Sequence of intervals between loss instants

$U(k) = E[S_n S_{n+k}]$: The correlation function for the interloss interval process.

Multiple Connection Models

Multiple Parallel TCP Connections



Main assumption: Session synchronization.

- The window size variations for all the K sessions happen together (i.e., their cycles start and end at the same time).
- This assumption is justified from observations that tail drop causes the packet loss to get synchronized across all sessions.
- This is because packets get dropped because of buffer overflow at the bottleneck node; hence, because typically multiple sessions encounter the buffer overflow condition at the same time, their window size evolution tends to get synchronized.

Parallel Connections with Equal Round Trip Latencies

Using synchronized connections assumption:

$$\frac{dW_i(t)}{dt} = \frac{dW_j(t)}{dt} \quad \forall i,j$$

From which it follows that

$$\frac{dW(t)}{dt} = \frac{K}{T} \quad \text{when } W(t) \leq CT$$

and

$$\frac{dW(t)}{dt} = \frac{KC}{W(t)} \quad \text{when } W(t) > \text{CT}$$

Since $R_i(t) = \frac{R(t)}{K} \quad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$ it is sufficient to find an expression for R(t)

Equal Round Trip Latencies

If B > CT, the bottleneck queue is always occupied so that

$$R(t) = C \quad \text{and} \quad R_i(t) = \frac{C}{K} \quad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$

If B < CT

$$R^{avg} = \frac{0.75}{\left[1 - \frac{\beta}{(1+\beta)^2}\right]} C,$$

and

$$R_i^{avg} = \frac{0.75}{\left[1 - \frac{\beta}{(1+\beta)^2}\right]} \frac{C}{K} \qquad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$

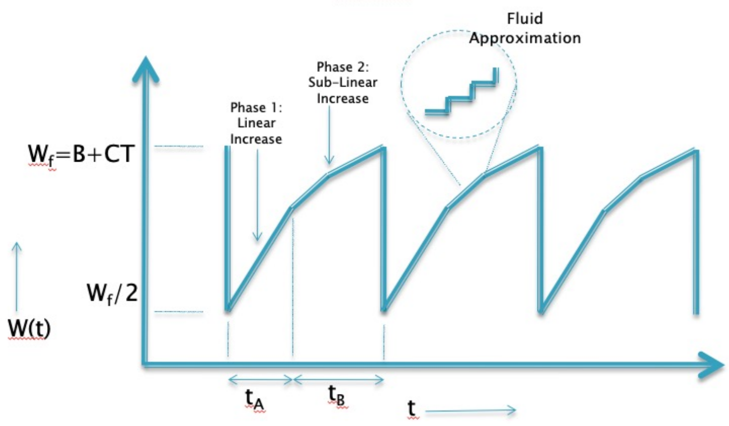
Parallel Connections with Un-Equal Round Trip Latencies

Note that

$$\frac{dW_i(t)}{dt} = \frac{1}{T_i + \frac{b(t)}{C}} \qquad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$

Also, the average throughput for the ith session is given by

$$R_i^{avg} = \frac{\frac{3}{8}(W_i^f)^2}{t_i^A + t_i^B} \qquad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$



Unequal Latencies: The Case of Large Buffers

Assume $B \gg CT_i,\ 1 = 1{,}2, \ldots, K$

This means that the bottleneck buffer is continuously occupied, so that

$$t_i^A = 0$$

This implies

$$\frac{dW_i(t)}{dt} \approx \frac{C}{b(t)}, \qquad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$

This implies that all the K windows increase at the same rate; hence, the maximum window sizes are also approximately equal, so that

$W_i^f = W_j^j$ *and* $t_i^B = t_j^B$ *for all* i, j

It follows that

$$R_i^{avg} \approx R_j^{avg} \qquad \mathrm{i,j} = 1, 2, \ldots, \mathrm{K}$$

Hence, when the number of buffers in the bottleneck node is large, each connection gets a fair share of the capacity despite any differences in propagation delays.

Unequal Latencies: The Case of Large Delay Bandwidth Product

Assume $B \ll CT_i,\ 1 = 1,2, \ldots, K$

Also assume synchronized connections

This implies $\frac{dW_i(t)}{dt} \approx \frac{1}{T_i}$

So that $\frac{W_i(t)}{W_j(t)} \approx \frac{T_j}{T_i}$ $\quad$ $\mathrm{i,j} = 1, 2, \ldots, \mathrm{K}$

Since

$$R_i(t) = \frac{W_i(t)}{T_i + \frac{b(t)}{C}} \qquad \mathrm{i} = 1, 2, \ldots, \mathrm{K}$$

It follows that

$$\frac{R_i(t)}{R_j(t)} \approx \frac{W_i/T_i}{W_j/T_j} \approx \left(\frac{T_j}{T_i}\right)^2 \qquad \mathrm{i,j} = 1, 2, \ldots, \mathrm{K}$$

Hence, for large delay-bandwidth products, TCP Reno with tail drop has a very significant bias against connections with larger propagation delays.

The Case of Large Delay Bandwidth Product

For more general AIMD protocols, the average Response Time is given by

$$R_{avg} = \frac{h}{T^e p^d}$$

h, e, d are constants

For example for TCP Cubic e = 0.25 and d = 0.75

For these systems it can be shown that

$$\frac{R^i_{avg}}{R^j_{avg}} = \left(\frac{T_j}{T_i}\right)^{\frac{e}{1-d}}$$

Using Random Drop Policies

- These results are critically dependent on the assumption that the packet drops are synchronized among all connections, which is true for tail-drop queues.
- A buffer management policy that breaks this synchronization leads to a fairer sharing of capacity among connections with unequal latencies.
- One of the objectives of RED buffer management was to break the synchronization in packet drops by randomizing individual packet drop decisions.
- Simulations have shown that RED indeed achieves this objective, and as a result, the ratio of the throughputs of TCP connections using RED varies as the first power of their round trip latencies rather than the square.

Further Reading

- Chapter 2 of Internet Congestion Control