Congestion Control in High Speed Networks: Part 1

Lecture 5

Subir Varma

Congestion Control Protocols

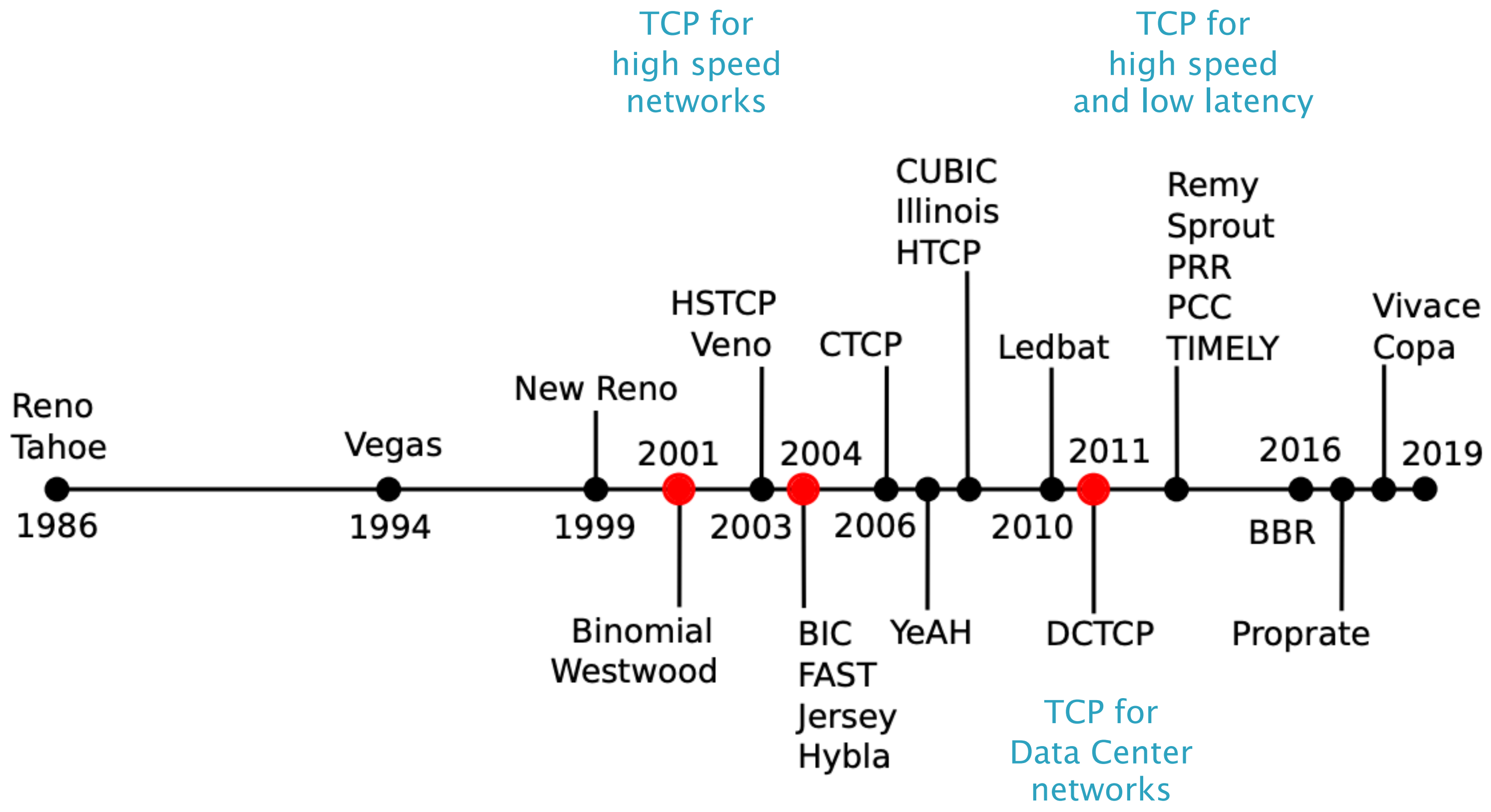


Fig. 1. The evolution of TCP congestion control.

"The Great Internet TCP Congestion Control Census" Mishra et.al. (2019)

Distribution of TCP Variants

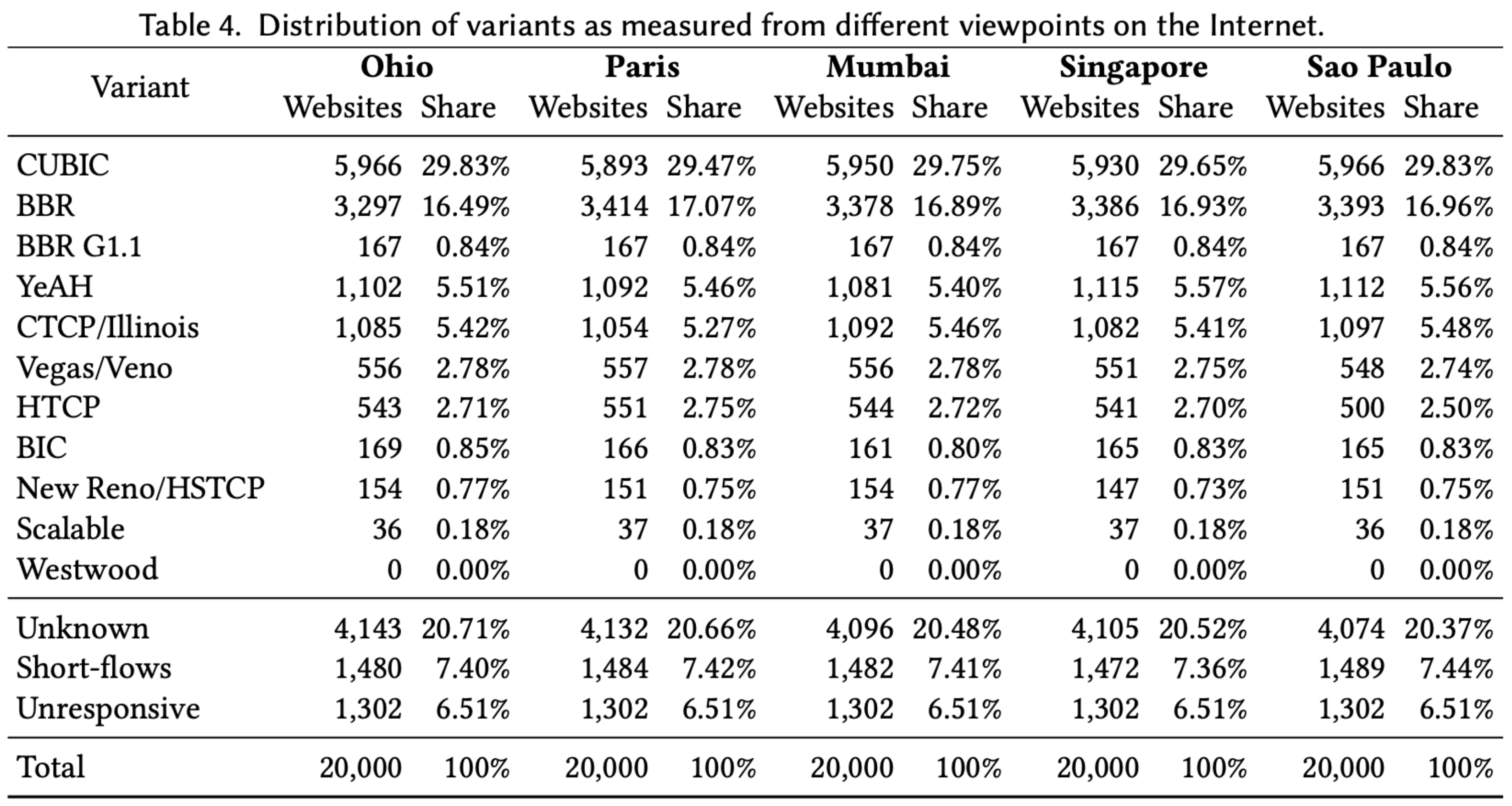


Table 4. Distribution of variants as measured from different viewpoints on the Internet.

Variant	**Ohio**		**Paris**		**Mumbai**		**Singapore**		**Sao Paulo**	
	Websites	Share	Websites	Share	Websites	Share	Websites	Share	Websites	Share
CUBIC	5,966	29.83%	5,893	29.47%	5,950	29.75%	5,930	29.65%	5,966	29.83%
BBR	3,297	16.49%	3,414	17.07%	3,378	16.89%	3,386	16.93%	3,393	16.96%
BBR G1.1	167	0.84%	167	0.84%	167	0.84%	167	0.84%	167	0.84%
YeAH	1,102	5.51%	1,092	5.46%	1,081	5.40%	1,115	5.57%	1,112	5.56%
CTCP/Illinois	1,085	5.42%	1,054	5.27%	1,092	5.46%	1,082	5.41%	1,097	5.48%
Vegas/Veno	556	2.78%	557	2.78%	556	2.78%	551	2.75%	548	2.74%
HTCP	543	2.71%	551	2.75%	544	2.72%	541	2.70%	500	2.50%
BIC	169	0.85%	166	0.83%	161	0.80%	165	0.83%	165	0.83%
New Reno/HSTCP	154	0.77%	151	0.75%	154	0.77%	147	0.73%	151	0.75%
Scalable	36	0.18%	37	0.18%	37	0.18%	37	0.18%	36	0.18%
Westwood	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%
Unknown	4,143	20.71%	4,132	20.66%	4,096	20.48%	4,105	20.52%	4,074	20.37%
Short-flows	1,480	7.40%	1,484	7.42%	1,482	7.41%	1,472	7.36%	1,489	7.44%
Unresponsive	1,302	6.51%	1,302	6.51%	1,302	6.51%	1,302	6.51%	1,302	6.51%
Total	20,000	100%	20,000	100%	20,000	100%	20,000	100%	20,000	100%

Problems with TCP Reno in High Speed Networks

- TCP Reno becomes unstable if either the link capacity, the propagation delay, or both become large. This results in severe oscillations of the transmission rates and buffer occupancies, which has been observed in practice in high-speed networks (see Lecture 3).

- Consider the following scenario: The link capacity is 10 Gbps, and the Round Trip Latency (RTT) is 100 ms, so the delay bandwidth product is about 100,000 packets (for a packet size of1250 bytes). For TCP to grow its window from the midpoint (after a multiplicative decrease) to full window size during the congestion avoidance phase, it will require 50,000 RTTs, which is about 5000 seconds (1.4 hours). If a connection finishes before then, the link will be severely underutilized.
  - Hence, when we encounter a combination of high capacity and large end-to-end delay, the multiplicative decrease policy on packet loss is too drastic, and the linear increase is not quick enough.

Problems (cont)

- The square-root formula for TCP throughput (see Chapter 2) shows that for a round trip latency of T, the packet drop rate has to be less than $1.5/(RT)^2$ to sustain an average throughput of R. Hence, using the same link and TCP parameters as above, the packet drop rate has to be less than 10^{-10} to sustain a throughput of 10 Gbps. This is much smaller than the drop rate that can be supported by most links.

New Algorithms for High Speed Links

- High Speed TCP (HSTCP)
- Binary Increase Control (BIC)
- CUBIC
  - Default in Linux Servers
  - 45% of servers in the Internet use either BIC or CUBIC. 30% still use TCP Reno.
- Compound TCP (CTCP)
  - Used in Windows Servers
  - 20% of servers use CTCP
- Yeah TCP
- BBR

Non TCP Algorithms

- XCP Express Control Protocol
- RCP Rate Control Protocol

These algorithms cannot be used over IP Networks since they require additional fields in the packet header

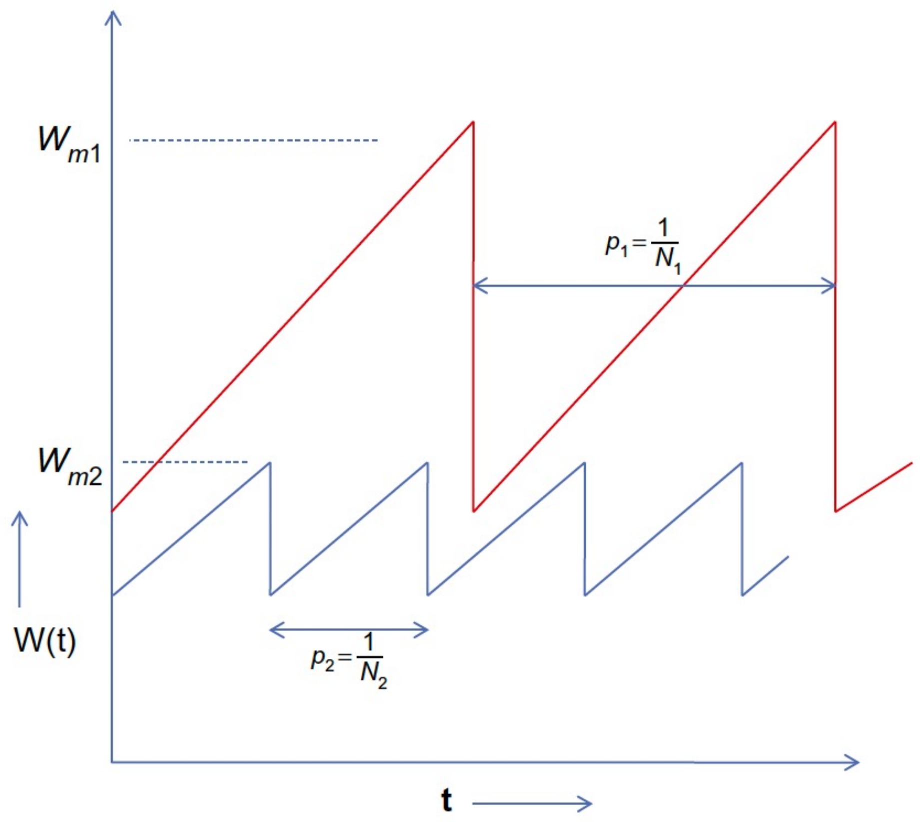
Requirements for a High Speed Protocol

1. It should be able to make efficient use of the high-speed link without requiring unrealistically low packet loss rates. This is not the case for TCP Reno, because of its conservative window increase and aggressive window decrease rules.

2. In case of very high-speed links, the requirement of inter-protocol fairness between the high speed TCP protocol and regular TCP is relaxed because regular TCP is not able to make full use of the available bandwidth. Hence, in this situation, it is acceptable for the high speed TCP variants to perform more aggressively than standard TCP.

3. If connections with different round trip latencies share a link, then they should exhibit good intra-protocol fairness.
   This should also be the case for connections with similar round trip latencies.

Lower p => Higher C



Assumption: All packet drops are due to buffer overflows.

Identical T
Different Capacities

$$p = \frac{1}{N} = \frac{8}{3W_m^2} \approx \frac{8}{3(CT)^2} \qquad \text{since} \qquad W_m \approx CT$$

The packet drop probability is inversely proportional to the square of the link speed

Response Function w

Definition $w = R_{avg} T$

Average Tpt

Minimum Round Trip Latency

w is the average number of packets transmitted per round trip time

Example Response Functions (Cont)

TCP Reno

$$w = \sqrt{\frac{3}{2p}} \quad or \quad \log w = 0.09 - 0.5 \log p$$

Additive increase/multiplicative decrease AIMD(32, 0.125)

$$R_{avg} = \frac{1}{T}\sqrt{\frac{a(2-b)}{2bp}} \qquad \log w = 1.19 - 0.5 \log p$$

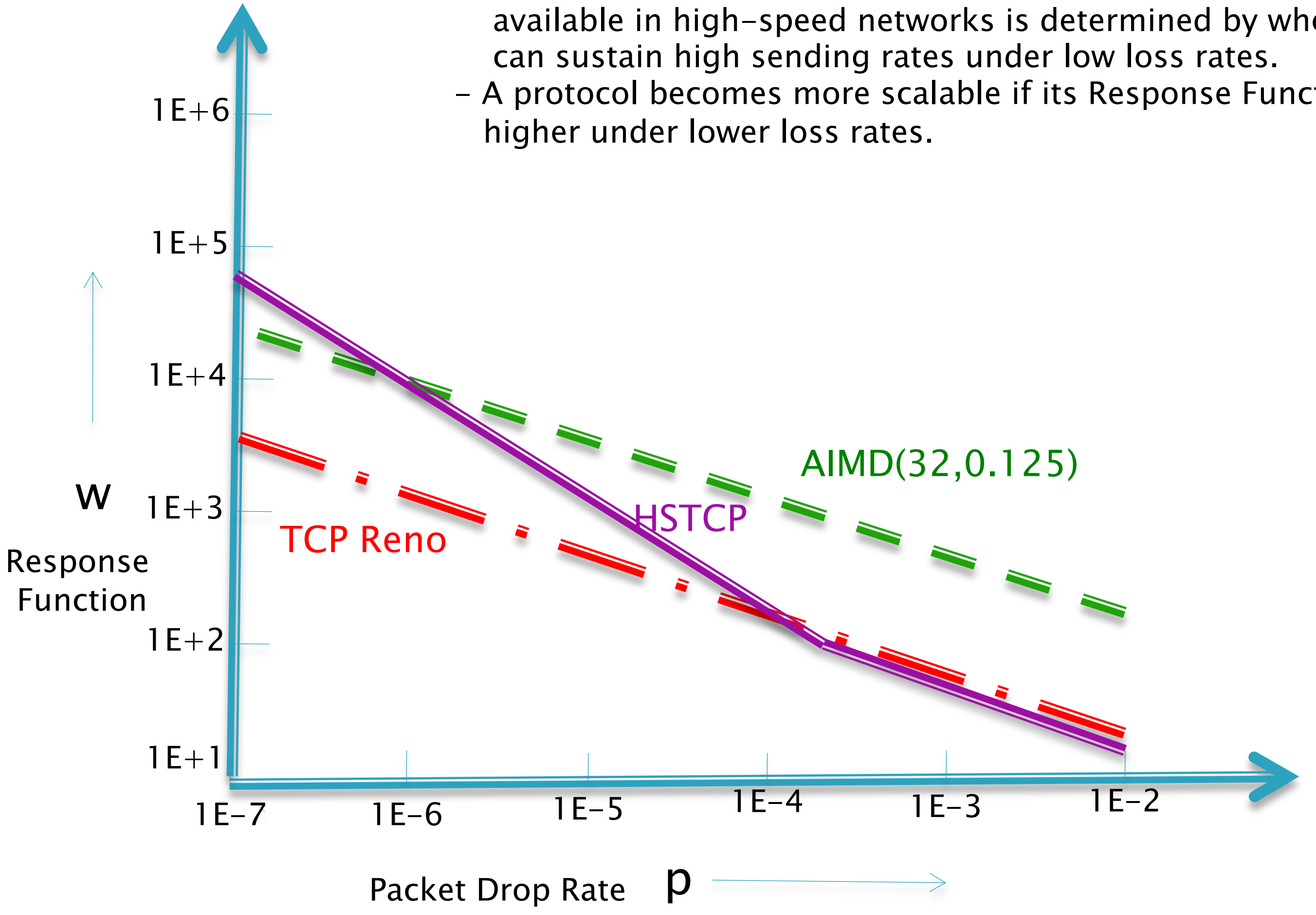
$$W \leftarrow \begin{cases} W + a & per \quad RTT \quad on \quad ACK \quad receipt \\ (1-b)W & on \quad packet \quad loss \end{cases}$$

High Speed TCP (HSTCP)

$$\log w = \begin{cases} -0.82 - 0.82 \log p & if \quad p < 0.0015 \\ 0.09 - 0.5 \log p & otherwise \end{cases}$$

Example Response Functions

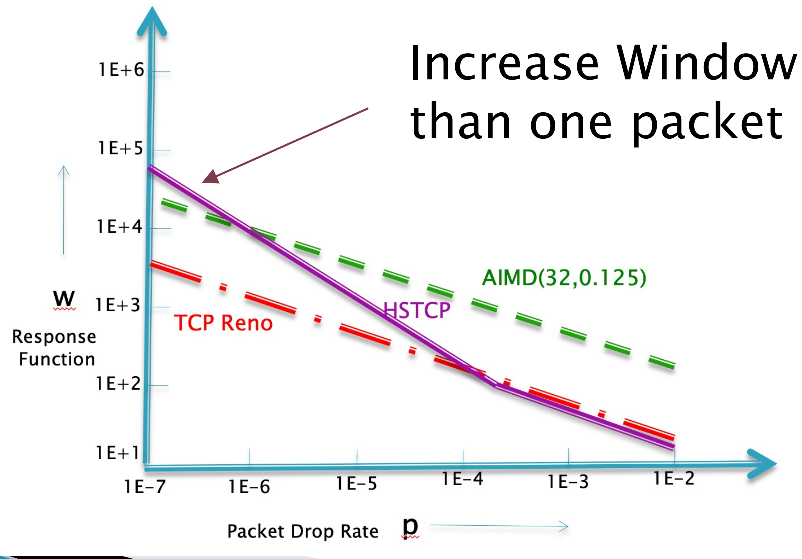
- The ability of a protocol to use the high amount of bandwidth available in high-speed networks is determined by whether it can sustain high sending rates under low loss rates.
- A protocol becomes more scalable if its Response Function is higher under lower loss rates.



Lower p corresponds to Higher Speed Links

Info from Response Function

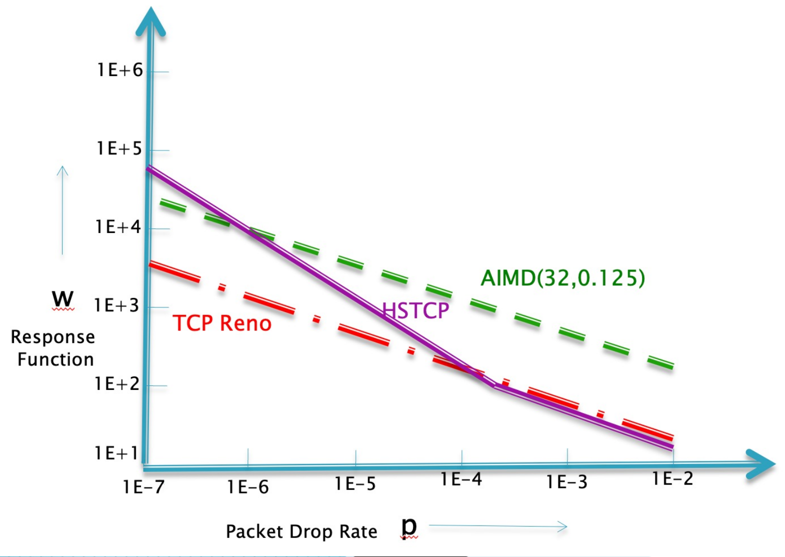
1. At small values of p (or equivalently for very high-speed links), it is desirable that the Response Function for the high-speed protocol be much larger than for TCP Reno.
   - For a given interval between packet drops, this allows the congestion window to grow to a much larger value compared with Reno.
   - If Reno and the high-speed protocol compete for bandwidth in a high-speed link, the latter will end up with a much larger portion of the bandwidth (i.e., the system will not be fair). However, this is deemed to be tolerable because the high-speed protocol is taking up link bandwidth that cannot be used by TCP Reno anyway.



Increase Window Size by more than one packet per RTT

Info from Response Function

2. Inter-Protocol Fairness: At larger values of p (or equivalently lower speed links), it is desirable for TCP friendliness that the point at which the two curves cross over lie as much to the left as possible.
   - The reason for this is that for lower speed links, the congestion control algorithm should be able to sustain smaller window sizes, which is better suited to coexisting with TCP Reno without taking away bandwidth from it.
   - To do this, the Response Function for the high-speed algorithm should be equal to or lower than that for TCP Reno after the two curves cross over.



- Revert back to TCP Reno for smaller w (HSTCP)
- Keep track of the connections' queue backlog in the network and switch to a less aggressive packet increment rule when the backlog exceeds a threshold (CTCP, TCP Africa).

Intra-Protocol Fairness with RTT

3. Intra-Protocol Fairness with different RTT:
   Assume that the throughput of a loss based congestion control protocol with constants A, e, d is given by

$$R = \frac{A}{T^e p^d} \quad \text{packets/s}$$

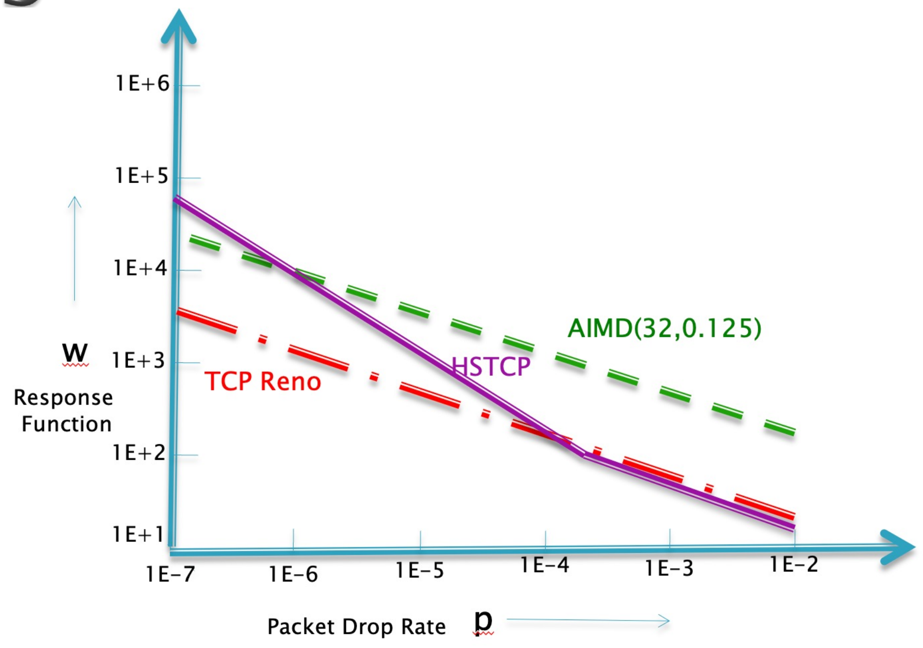
Then the throughput ratio between two flows with round trip latencies T1 and T2, whose packet losses are svnchronized, is given by

$$\frac{R_1}{R_2} = \left(\frac{T_2}{T_1}\right)^{\frac{e}{1-d}}$$

As d increases, the slope of the response function and RTT unfairness both increase => The slope of the response function on the log-log plot determines its intra-protocol RTT unfairness.

All three properties of the High Speed Protocol can be read off from the Response Function

Inter-Protocol vs Intra-Protocol Fairness



There is a tradeoff between requirements 2 and 3 as follows: Even though it is possible to design a more TCP-friendly protocol by moving the point of intersection with the TCP curve to lower packet drop rate, this leads to an increase in the slope of the Response Function line, thus hurting RTT fairness.

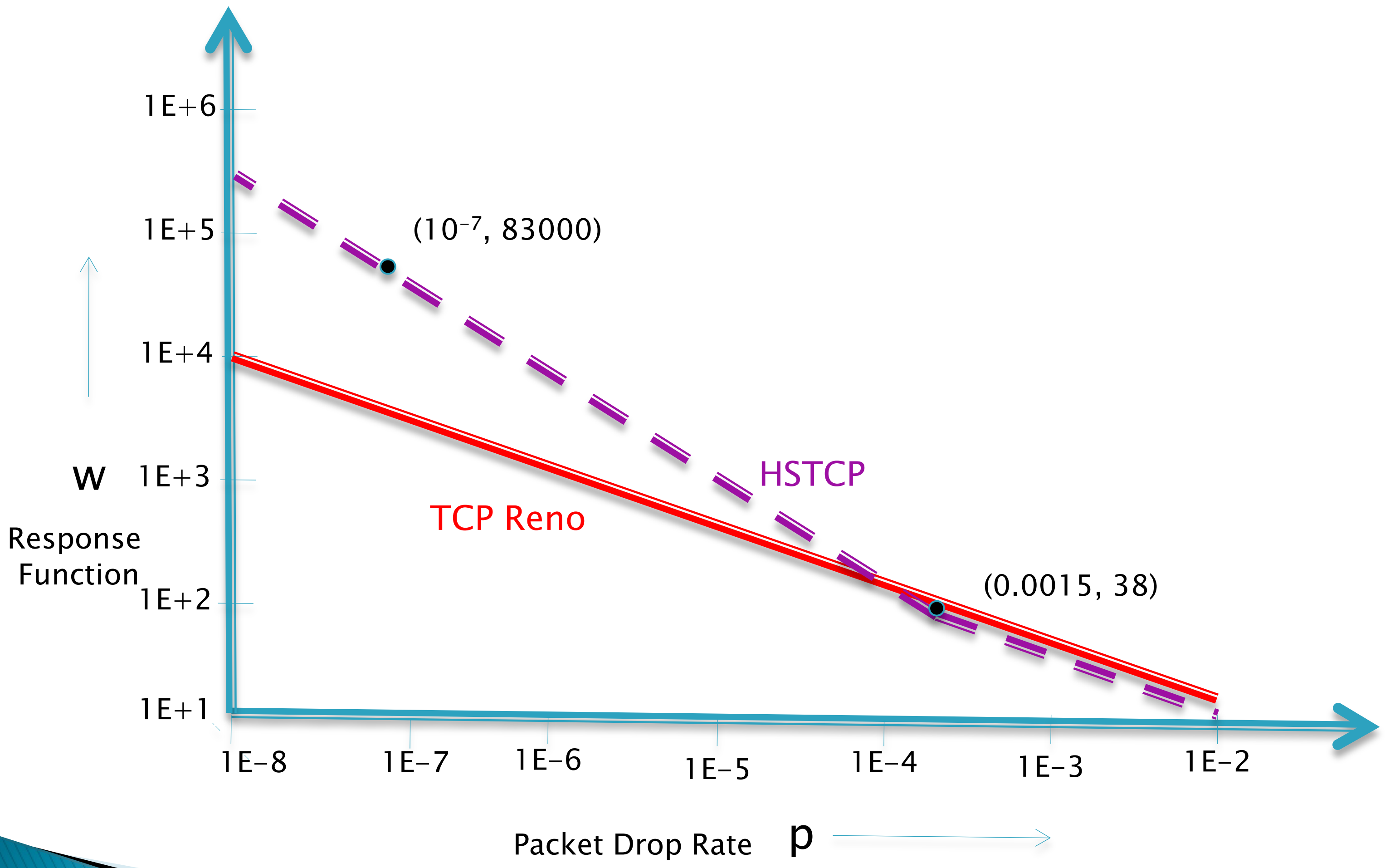
HSTCP
High Speed Transport Control Protocol

HSTCP Objectives

- The HSTCP Response Function should coincide with that of TCP Reno when the packet drop rate exceeds some threshold P (or equivalently when the Response Function is equal to or less than w packets). As a default HSTCP chooses $P_0 = 0.0015$ and $w_0 = 538$ packets.

- We next choose a large ppr, say, $w_1 = 583{,}000$ packets (which corresponds to a link capacity of 10 Gbps for packet size of 1500 bytes and round trip time $T = 100$ ms) and a corresponding packet loss rate of $P_1 = 10^{-7}$. For packet loss rates less than P_0, we specify that the HSTCP Response Function should pass through the points (P_0, w_0) and (P_1, w_1) and should also be linear on a log-log scale

High Speed (HSTCP) Protocol



HSTCP Analysis: Response Function

The Response Function for HSTCP is given by

$$\log w = S(\log p - \log P_0) + \log w_0$$

Where

$$S = \frac{\log w_1 - \log w_0}{\log P_1 - \log P_0}$$

This equation can also be written as

$$w = w_0 \left(\frac{p}{P_0}\right)^S \quad \text{packets.}$$

If we substitute the values of (P_0,w_0) and (P_1,w_1) in Response Time equation, then we obtain S = -0.82, so that the HSTCP Response Function is given by

$$w = \frac{0.15}{p^{0.82}} \quad \text{packets for } p \leq P_0.$$

AIMD Equation for HSTCP

$$ACK : W \leftarrow W + \frac{a(W)}{W}$$

$$Drop : W \leftarrow W - b(W)W$$

Find the functions a(W) and b(W)

For the case $W \leq w_0$, HSTCP defaults to TCP Reno, so that

$$a(W) = 1 \; and \; b(W) = 0.5 \; for \; W \leq w_0$$

For $W > w_{0,}$ we assume that b(W) varies linearly with respect to log(W) for $W \in [w_0, w_1]$ with $b(w_0) = 0.5$ and $b(w_1) = B$, where B is target maximum value for b(W) (defaulted to B = 0.1 at $W = w_1$)

$$b(W) = (B - 0.5)\frac{\log W - \log w_0}{\log w_1 - \log w_0} + 0.5$$

AIMD Equation for HSTCP (cont)

$$ACK : W \leftarrow W + \frac{a(W)}{W}$$

$$Drop : W \leftarrow W - b(W)W$$

Find the functions a(W) and b(W)

To obtain the corresponding a(W), we use the following formula for the response function of AIMD congestion control: We choose a point $W \in [w_0, w_1]$ and compute the corresponding probability P(W) . Substitute (P(W),W) into the following equation

$$W = \sqrt{\frac{a(W)(2-b(W))}{2P(W)b(W)}}$$

so that

$$a(W) = P(W)W^2 \frac{2b(W)}{2-b(W)}$$

Note that there are a couple of heuristics in the use of equation 12:
(1) equation 12 is a formula for the response time function, not the window size, and
(2) equation 12 was derived for the case when a and b are constants, which is not the case here.

AIMD Equation for HSTCP (cont)

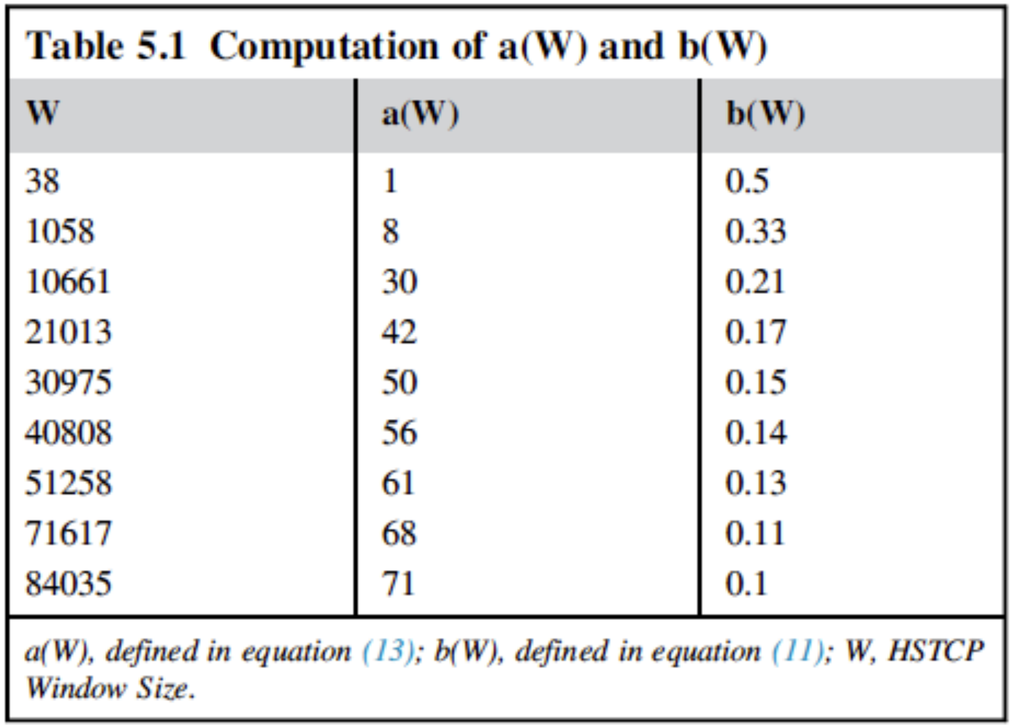


Table 5.1 Computation of a(W) and b(W)

W	a(W)	b(W)
38	1	0.5
1058	8	0.33
10661	30	0.21
21013	42	0.17
30975	50	0.15
40808	56	0.14
51258	61	0.13
71617	68	0.11
84035	71	0.1

a(W), defined in equation (13); b(W), defined in equation (11); W, HSTCP Window Size.

HSTCP Properties

A significant issue with HSTCP is the fact that the window increment function
a(W) increases as W increases and exceeds more 70 for large values of W.

This implies that at the time when the link buffer approaches capacity, it is subject
to a burst of more than 70 packets, all of which can then be dropped.
Almost all follow-on high-speed design reduce a(W) as the link approaches saturation.

HSTCP is more stable than TCP Reno at high speeds because it is less aggressive in reducing
its window after a packet loss which reduces the queue size oscillations at the bottleneck node.

HSTCP Intra-Protocol Fairness with RTT

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

$$\frac{R_1}{R_2} = \left(\frac{T_2}{T_1}\right)^{\frac{e}{1-d}}$$

If we apply equation 4 to HSTCP, then substituting e = 1, d = 0.82, we obtain

$$\frac{R_1}{R_2} = \left(\frac{T_2}{T_1}\right)^{\frac{1}{1-d}} = \left(\frac{T_2}{T_1}\right)^{5.55}$$

- Because of the aggressive window increment policy of HSTCP, combined with small values of the decrement multiplier, connections with a larger round trip latency are at a severe throughput disadvantage.

- There is a positive feedback loop which causes the connection with the large latency to loose throughput, which operates as follows:

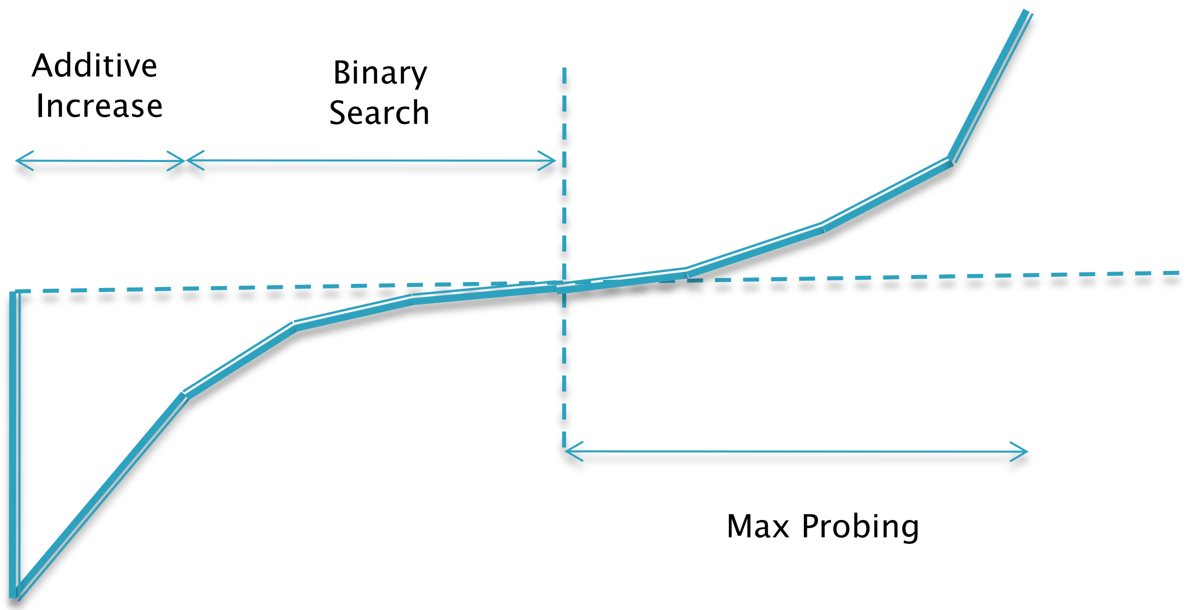
  - Even if the two connections start with the same window size, the connection with smaller latency will gain a small advantage after a few round trip delays because it is able to increase its window faster.
  - However this causes its window size to increase, which further accelerates this trend.
  - Conversely the window size of the large latency connection goes into a downward spiral as reductions in its throughput causes a further reduction in its window size.

TCP BIC
Binary Increase Protocol

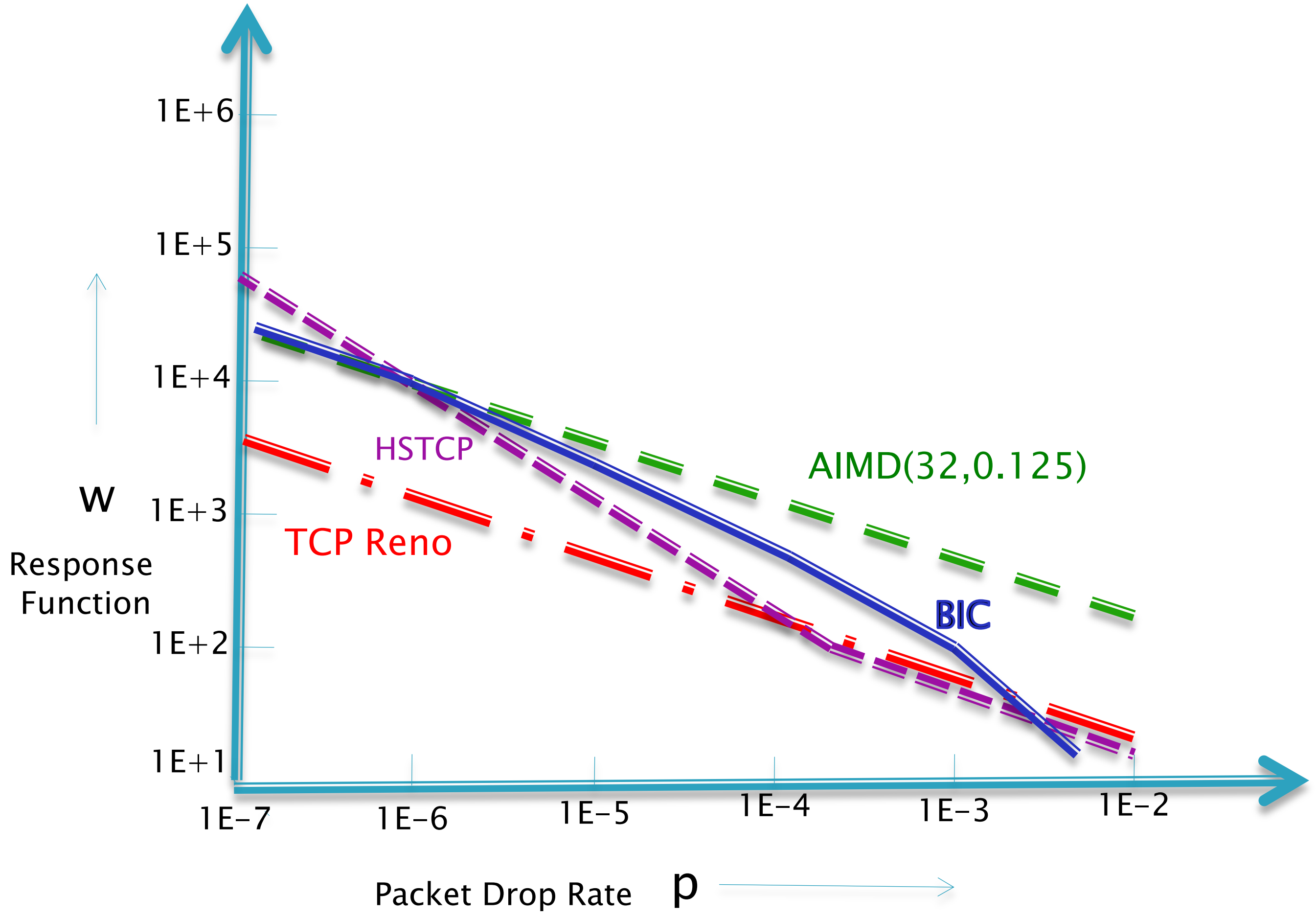
TCP BIC

- The main idea behind TCP BIC, is to grow the TCP window quickly when the current window size is far from the link saturation point (at which the previous loss happened), and if it is close to the saturation point then the rate of increase is slowed down.
- This results in a concave increase function, which decrements the rate of increase as the window size increases.
- The small increments result in a smaller number of packet losses if the window size exceeds the capacity. This makes BIC (and its successor CUBIC) very stable and also very scalable.
- If the available link capacity has increased since the last loss, then both BIC and CUBIC increase the window size exponentially using a convex function. Since this function shows sub-linear increase initially, this also adds to the stability of the algorithm for cases in which the new link capacity did not increase significantly.

Window Growth Function of TCP BIC



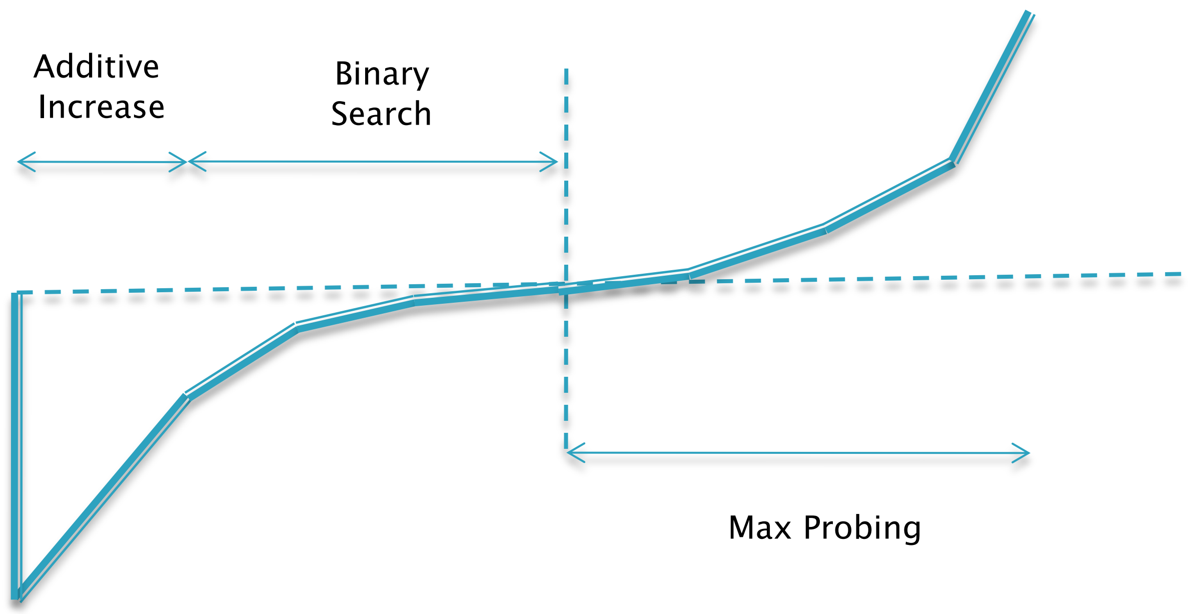
BIC Response Function



Observations from BIC Response Function

- Because the BIC response time graph intersects with that of Reno around the same point as that of HSTCP, it follows BIC is also TCP friendly at lower link speeds.
- The slope of BIC's response time at first rises steeply and then flattens out to that of TCP Reno for larger link speeds. This implies that for high-speed links, BIC exhibits the RTT intra-protocol fairness property (or it is at least as fair as TCP Reno).
- Because of the fast initial rise in its response time, BIC scales well with link speeds.

BIC Window Growth: Three Phases



1. Additive Increase
2. Binary Search Increase
3. Max Probing

Binary Search Increase

The Binary Search window increase rule in BIC was inspired by the binary search algorithm and operates as follows:

- If W_{max} is the maximum window size that was reached just before the last packet loss, and W_{min} is the window size just after the packet loss, then after 1 RTT, the algorithm computes W_{mid}, the midpoint between W_{max} and W_{min}, and sets the current window size cwnd to W_{mid}.
- After the resulting packet transmissions:
  - If there are no packet losses, then W_{min} is set to W_{mid}, or
  - If there are packet losses then W_{max} is set to W_{mid}.
- For the case of no packet losses the process repeats for every RTT until the difference between W_{max} and W_{min} falls below a preset threshold S_{min}. Results in logarithmic increase in window size.

Binary Search Increase (cont)

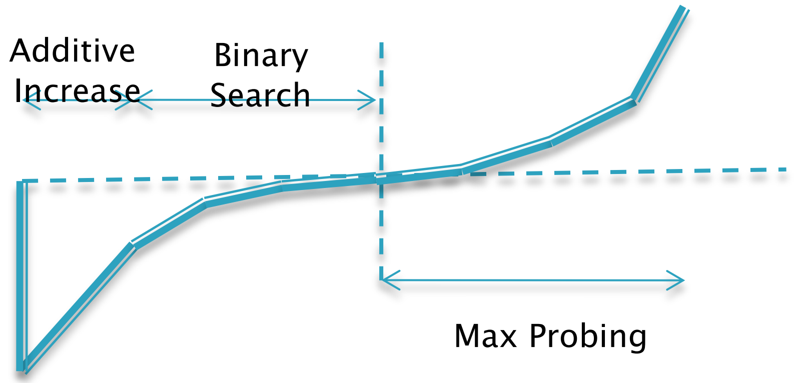
Binary Search allows the bandwidth probing to be more aggressive when the difference between the current window cwnd and the target window W_{max} is large and gradually becomes less aggressive as cwnd gets closer to W_{max}.

This results in a reduction in the number of lost packets as the saturation point is reached.

This behavior contrasts with HSTCP, which increases its window increase rate near the link saturation point, resulting in excessive packet loss.

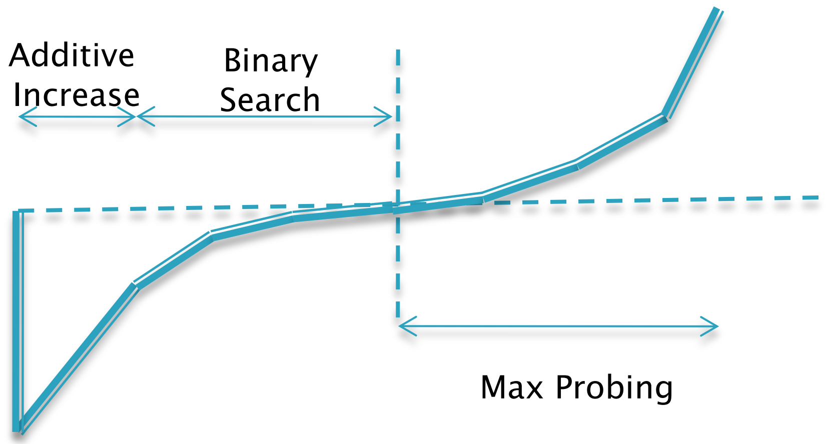
Additive Increase

- When the distance to W_{mid} from the current W_{min} is large, then increasing the window cwnd to that midpoint leads to a large burst of packets transmitted into the network, which can result in losses.
- In this situation, cwnd is increased by a configured maximum step value S_{max} and W_{min} is set cwnd.
- This continues until the distance between W_{min} and W_{mid} falls below S_{max}, at which time W_{min} is set directly to W_{mid}.
- After a large window size reduction, the additive increase rule leads to an initial linear increase in window size followed by a logarithmic increase for the last few RTTs.



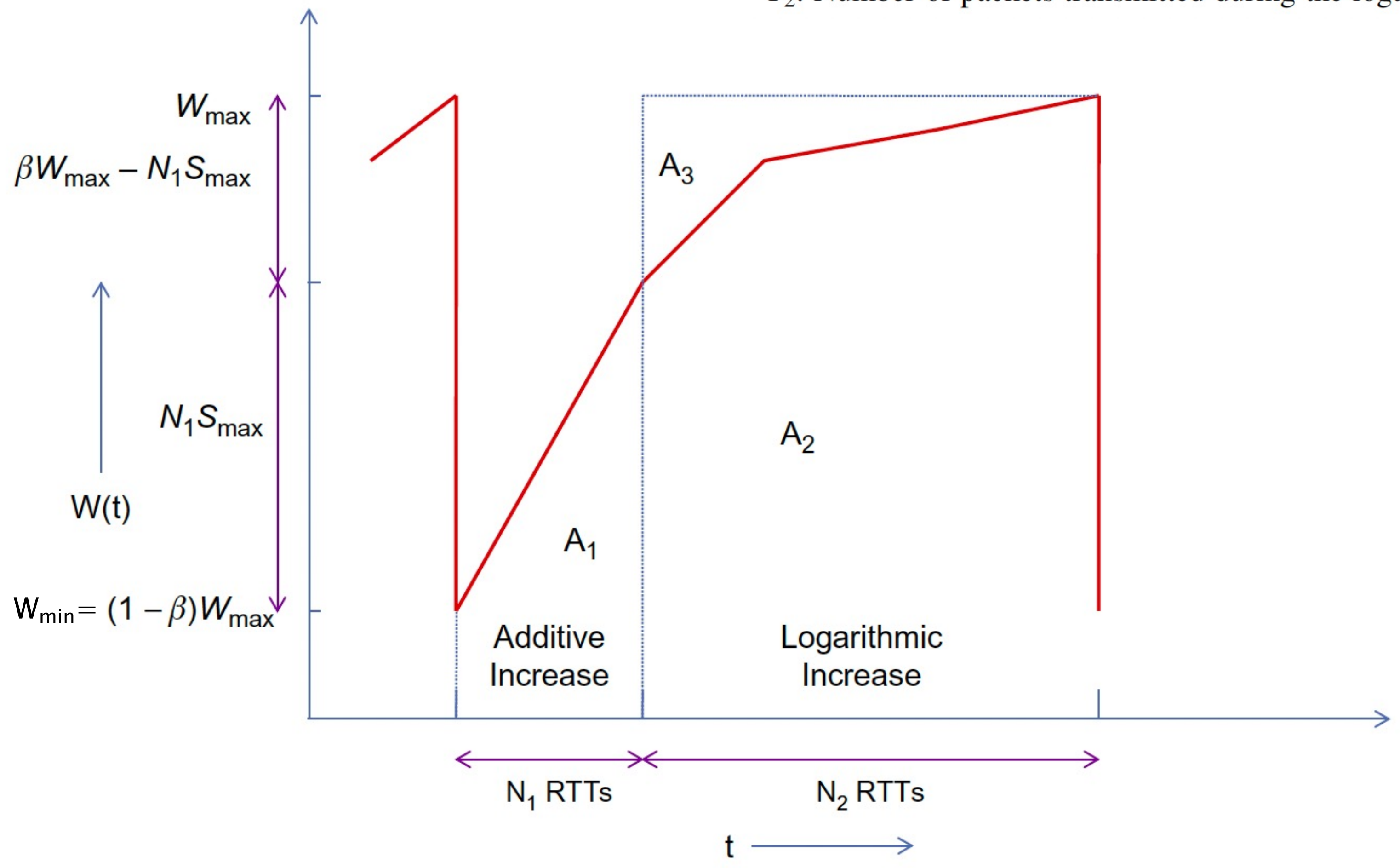
Max Probing

- When the current window size grows past W_{max}, the BIC algorithm switches to probing for the new maximum window, which is not known.
- It does so in a slow-start fashion by increasing its window size in the following sequence for each RTT: $W_{max} + S_{min}$, $W_{max} + 2S_{min}$, $W_{max} + 4S_{min}$, $W_{max} + S_{max}$.
- The reasoning behind this policy is that it is likely that the new saturation point is close to the old point; hence, it makes sense to initially gently probe for available bandwidth before going at full blast.
- After the max probing phase, BIC switches to additive increase using the parameter S_{max}.



BIC Analysis

W_{max}: Maximum window size
W_{min}: Minimum window size
N_1: Number of RTT rounds in the additive increase phase
N_2: Number of RTT rounds in the logarithmic increase phase
Y_1: Number of packets transmitted during the additive increase phase
Y_2: Number of packets transmitted during the logarithmic increase phase



BIC Analysis

As per the deterministic approximation technique, we make the assumption that the number of packets sent in each cycle of the congestion window is fixed (and equal to 1/p, where p is the packet drop rate). The average throughput R_{avg} is then given by

$$R_{\text{avg}} = \frac{Y_1 + Y_2}{T(N_1 + N_2)} = \frac{1/p}{T(N_1 + N_2)} \tag{15}$$

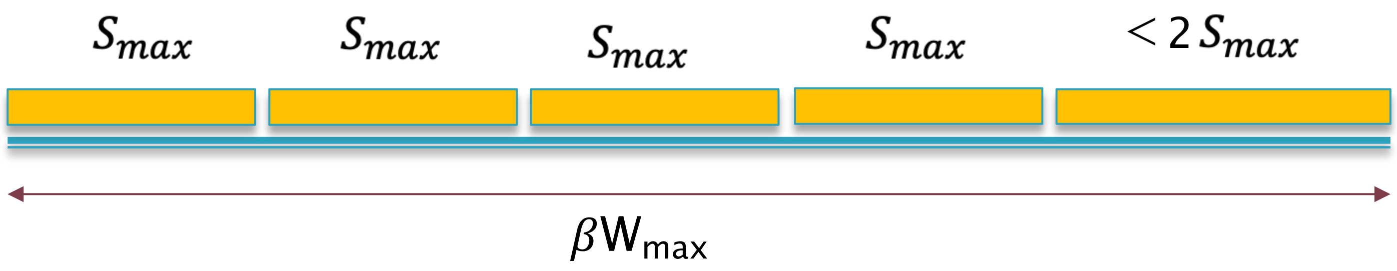
BIC Analysis: N1

N_1: Number of RTT rounds in the additive increase phase

BIC switches to the logarithmic increase phase when the distance from the W_{min} to W_{max} is less than $2S_{max}$

Proof:
Additive increase until the distance between W_{min} and W_{mid} falls below S_{max} i.e.,

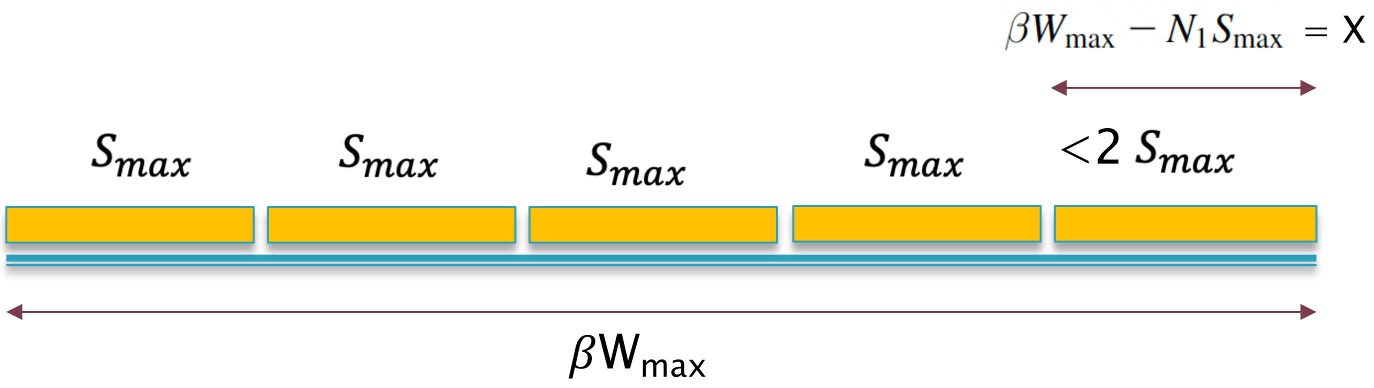
$$\frac{W_{max} + W_{min}}{2} - W_{min} < S_{max}$$



$$N_1 = \max\left(\left\lceil \frac{\beta W_{max}}{S_{max}} \right\rceil - 2, 0\right)$$

N_2: Number of RTT rounds in the logarithmic increase phase

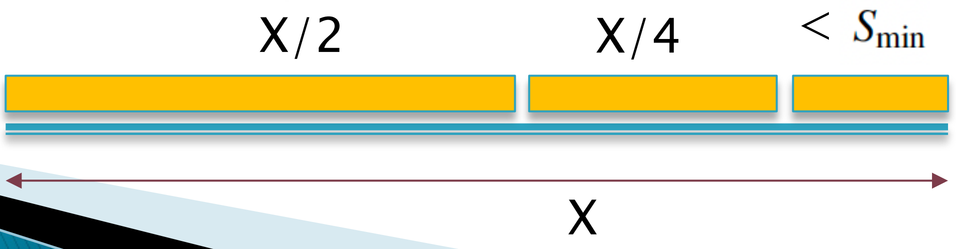
BIC Analysis: N_2



$$\frac{X}{2} + \frac{X}{2^2} + \cdots + \frac{X}{2^{N_2}} = X - S_{\min}$$

=> $$X\left(1 - \frac{1}{2^{N_2}}\right) = X - S_{\min}, \text{ so that}$$

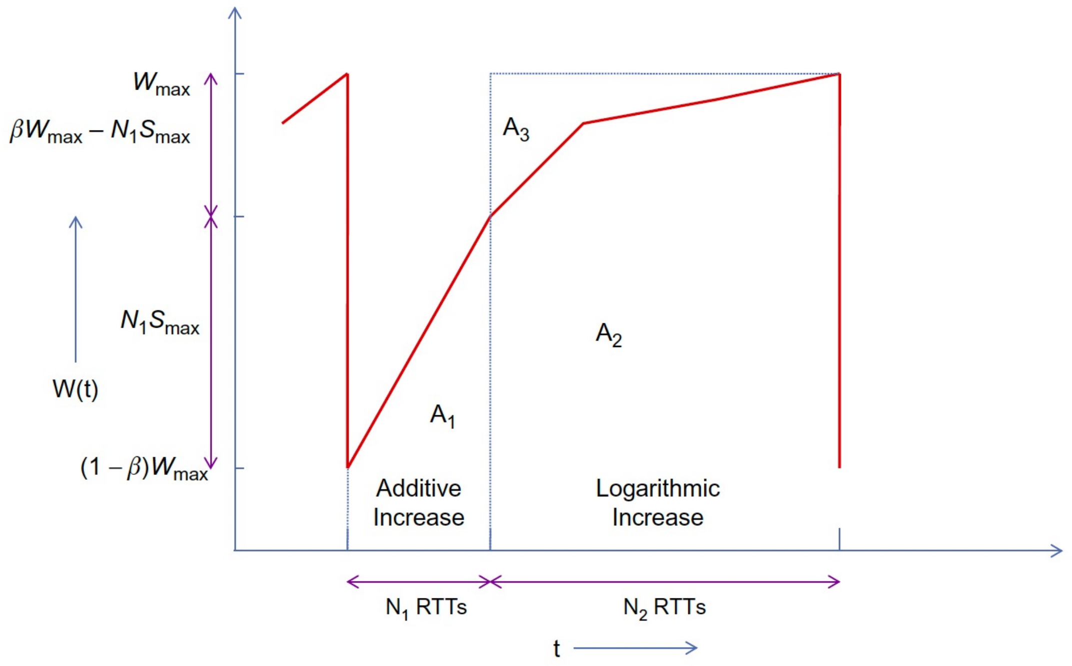
$$N_2 = \log\left(\frac{\beta W_{\max} - N_1 S_{\max}}{S_{\min}}\right) + 2$$



Binary Increase

Y_1: Number of packets transmitted in additive increase phase

BIC Analysis: Y_1



$$Y_1 = \frac{1}{T}\int_0^{N_1 T} W(T)dt \;=\frac{A_1}{T} \tag{20}$$

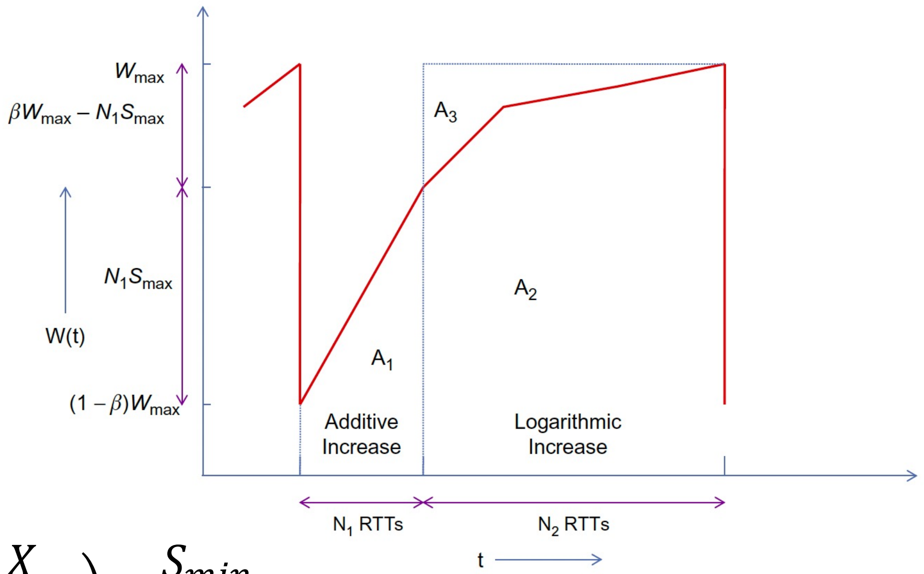
The integral is given by the area A_1 under the W(t) curve for the additive increase phase (Figure 5.6), which is given by

$$A_1 = (1-\beta)W_{\max}N_1T + \frac{1}{2}(N_1-1)S_{\max}N_1T, \text{ so that}$$

$$Y_1 = (1-\beta)W_{\max}N_1 + \frac{1}{2}(N_1-1)S_{\max}N_1 \tag{21}$$

Y_2: Number of packets transmitted in logarithmic increase phase

BIC Analysis: Y_2



$$Y_2 = \frac{1}{T}\int_{N_1T}^{N_2T} W(t)dt = \frac{A_2}{T} = \frac{W_{max}N_2T - A_3}{T}$$

$$\frac{A_3}{T} = \left(\frac{X}{2}+\frac{X}{2^2}\right) + \left(\frac{X}{2^2}+\frac{X}{2^3}\right) + \cdots + \left(\frac{X}{2^{N_2}}+\frac{X}{2^{N_2+1}}\right) + \frac{S_{min}}{2}$$

$$= X\left(1-\frac{1}{2^{N_2}}\right) + \frac{X}{2}\left(1-\frac{1}{2^{N_2}}\right) + \frac{S_{min}}{2}$$

$$= 1.5X - S_{min}$$

$$\boxed{Y_2 = W_{max}N_2 - 1.5(\beta W_{max} - N_1 S_{max}) + S_{min}}$$

BIC Analysis

$$R_{\text{avg}} = \frac{Y_1 + Y_2}{T(N_1 + N_2)} = \frac{1/p}{T(N_1 + N_2)}$$

Use $\frac{1}{p} = Y_1 + Y_2$ to express W_{max} as a function of p, β, S_{max}, S_{min}

$$N_1 = \max\left(\left\lceil \frac{\beta W_{\max}}{S_{\max}} \right\rceil - 2, 0\right) \qquad N_2 = \log\left(\frac{\beta W_{\max} - N_1 S_{\max}}{S_{\min}}\right) + 2$$

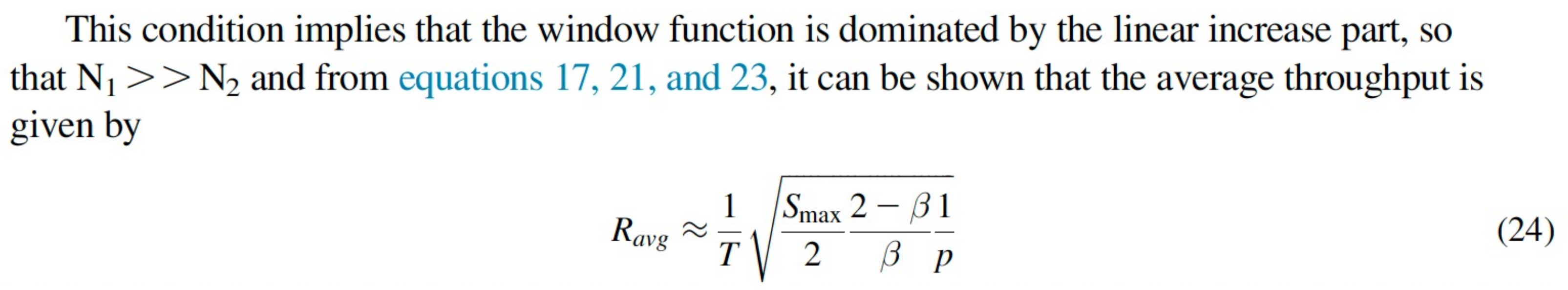
$$\frac{1}{p} = (1-\beta) W_{\max} N_1 + \frac{1}{2}(N_1 - 1) S_{\max} N_1 + W_{max} N_2 - 1.5(\beta W_{max} - N_1 S_{max}) + S_{min}$$

A closed form expression for W_{max} does not exist in general

BIC Analysis

Special Cases:

(1) $\beta W_{max} \gg 2S_{max}$:



This condition implies that the window function is dominated by the linear increase part, so that $N_1 >> N_2$ and from equations 17, 21, and 23, it can be shown that the average throughput is given by

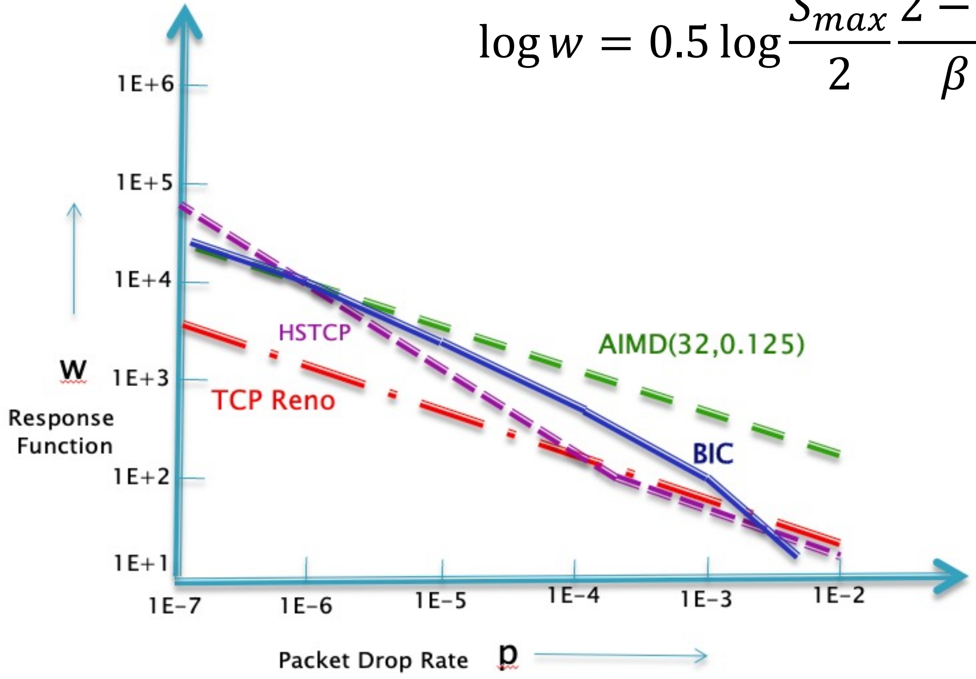
$$R_{avg} \approx \frac{1}{T}\sqrt{\frac{S_{\max}}{2}\frac{2-\beta}{\beta}\frac{1}{p}} \qquad (24)$$

For large window sizes (i.e. very high speed links, BIC reduces to AIMD congestion control

Shape of the BIC Response Time Curve

Because large values of W_{max} also correspond to large link capacities (because $W_{max} \approx CT$), it follows that for high-capacity links, BIC operates similar to an AIMD protocol with increase parameter $a = S_{max}$, decrease parameter of $b = \beta$, and the exponent $d = 0.5$. This follows from the fact that for high-capacity links, the BIC window spends most of its time in the linear increase portion.

$$\log w = 0.5 \log \frac{S_{max}}{2}\frac{2-\beta}{\beta} - 0.5 \log p$$



BIC Analysis: Special Cases

2. $\beta W_{\max} > 2S_{\max}$ and $\beta W_{\max}$ is divisible by $S_{\max}$.
From equations 21 to 23, it follows that

$$W_{\max} = \frac{-b + \sqrt{b^2 + 4a\left(c + \frac{1}{p}\right)}}{2a}$$

where

$$a = \frac{\beta(2-\beta)}{2S_{\max}} \quad b = \log\frac{S_{\max}}{S_{\min}} + \frac{2-\beta}{\beta} \quad c = S_{\max} - S_{\min}$$

and the throughput is given by

$$R_{avg} = \frac{1}{T}\frac{2-\beta}{p}\frac{1}{\sqrt{b^2 + 4a\left(c + \frac{1}{p}\right)} + (1-\beta)b + \frac{\beta(2-\beta)}{2}}$$

Shape of the BIC Response Time Curve

For moderate values of C, we can get some insight into BIC’s behavior by computing the constants in equation for R_{avg}. If we choose the following values for BIC’s parameters: $\beta = 0.125$; $S_{max} = 32$; Smin = 0.01. It follows that a = 0.0036, b = 18.5, and c = 31. It follows that

$$R_{avg} = \frac{1.875}{T}\sqrt{\frac{1}{359.02p^2 + 0.014p}}$$

This formula implies that if $359p^2 >> 0.014p$, i.e., $p >> 3.9E(-5)$, then it follows that

$$R_{avg} = \frac{0.1}{Tp}$$

Hence, it follows that for larger packet drop rates (i.e., smaller values of C), the exponent d = 1.

Because d = 0.5 for large link capacities and d = 1 for smaller link capacities, it follows that for intermediate value of capacity $0.5 < d < 1$, which explains the concave shape of the BIC response function Figure 5.4.

$$R_{avg} = \frac{1}{T}\frac{2-\beta}{p}\frac{1}{\sqrt{b^2 + 4a\left(c + \frac{1}{p}\right) + (1-\beta)b + \frac{\beta(2-\beta)}{2}}}$$

RTT Fairness for BIC

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

$$\frac{R_1}{R_2} = \left(\frac{T_2}{T_1}\right)^{\frac{e}{1-d}}$$

$$R_{avg} \approx \frac{1}{T}\sqrt{\frac{S_{\max}}{2}\frac{2-\beta}{\beta}\frac{1}{p}}$$

For large window sizes, d = 0.5

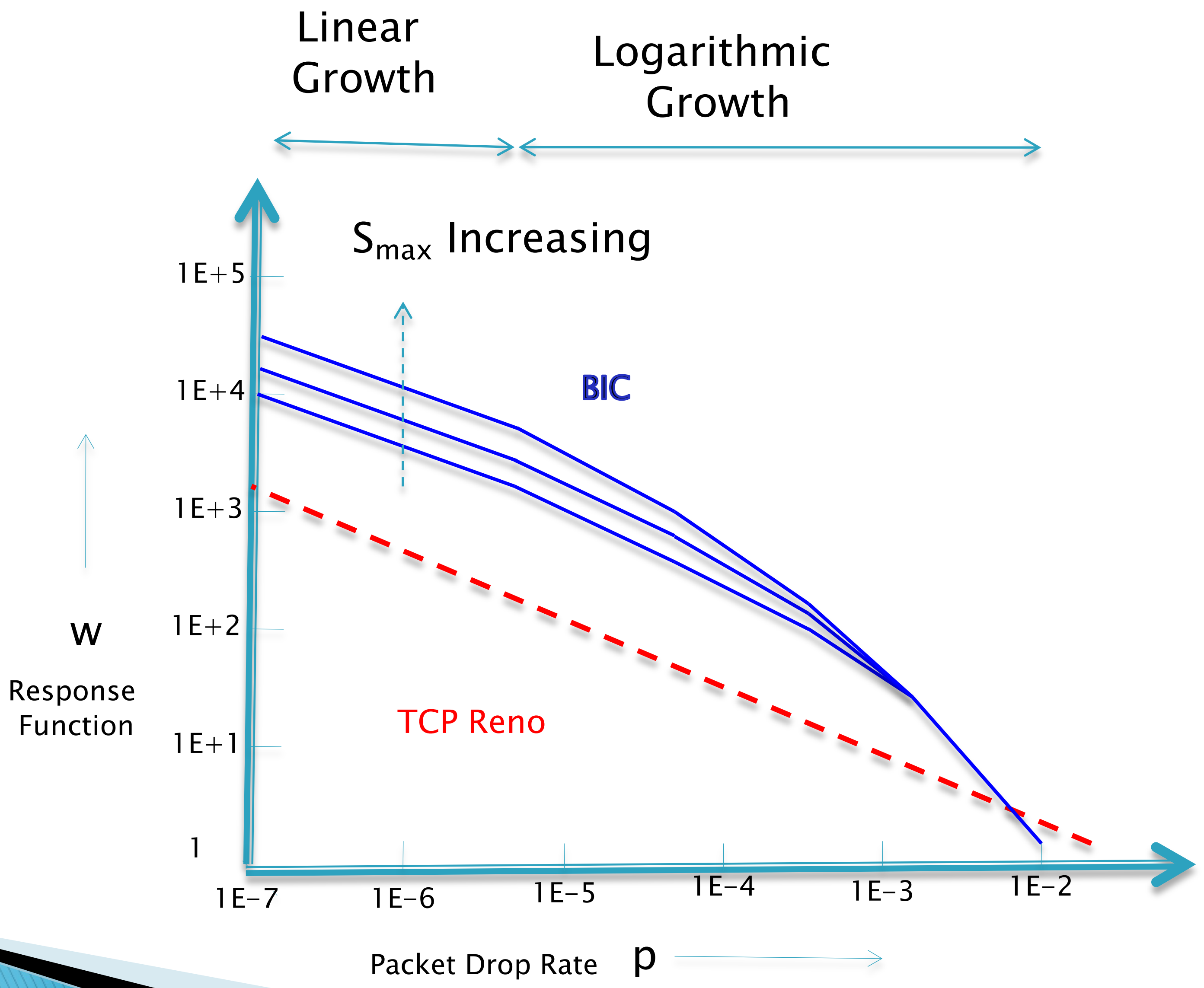
$$R_{avg} = \frac{1.875}{T}\sqrt{\frac{1}{359.02p^2 + 0.014p}}$$

For moderate window sizes, d =1

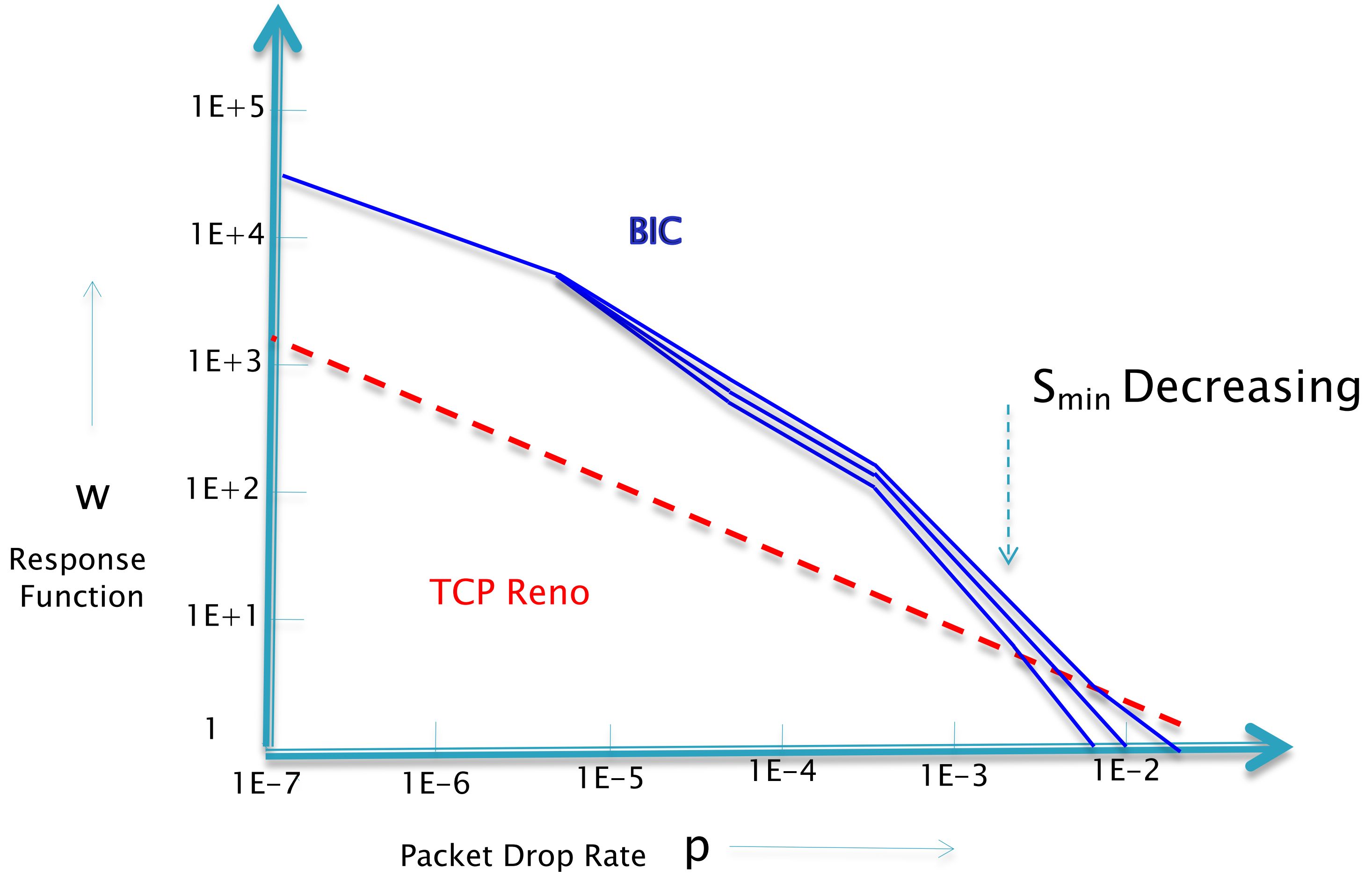
For high BW links, BIC has the same RTT fairness as Reno

For moderate BW links, BIC can be very unfair to connections with large RTT.

BIC Response Function with Varying S_{max}



BIC Response Function with Varying S_{min}



Reducing S_{min} makes BIC more TCP friendly, but also makes the RTT unfairness worse.

Further Reading

- Chapter 5, Sections 5.1–5.4 of Internet Congestion Control