TCP Response Under Loss Conditions

It is useful to understand TCP's behavior and in particular, TCP's behavior in the presence of loss. This may help dispell some of the myth regarding emminent collapse of the Internet. This also provides background needed to understand how an application's behavior can affect their own performance, why variations in TCP's algorithms can dramaticly affect performance and how network equipment can affect TCP's performance.

presented by Curtis Villamizar (ANS)

ftp://engr.ans.net/pub/slides/nanog/feb-1997/tcp-congested.ps

TCP References

- RFCs
- numerous acedemic papers
- BSD 4.4 Lite - reference code base
- TCP Illustrated - W. R. Stevens

RFCs alone don't tell the whole story.

A Rough TCP Timeline

1984	base TCP RFC
1988	congestion collapse predicted and observed
1988	slow start and congestion avoidance
1990	fast retransmit (TCP Tahoe)
1992	fast recovery (TCP Reno)
1993–5	window scaling implementations
1996–7	SACK implementations

Stages of a TCP Connection

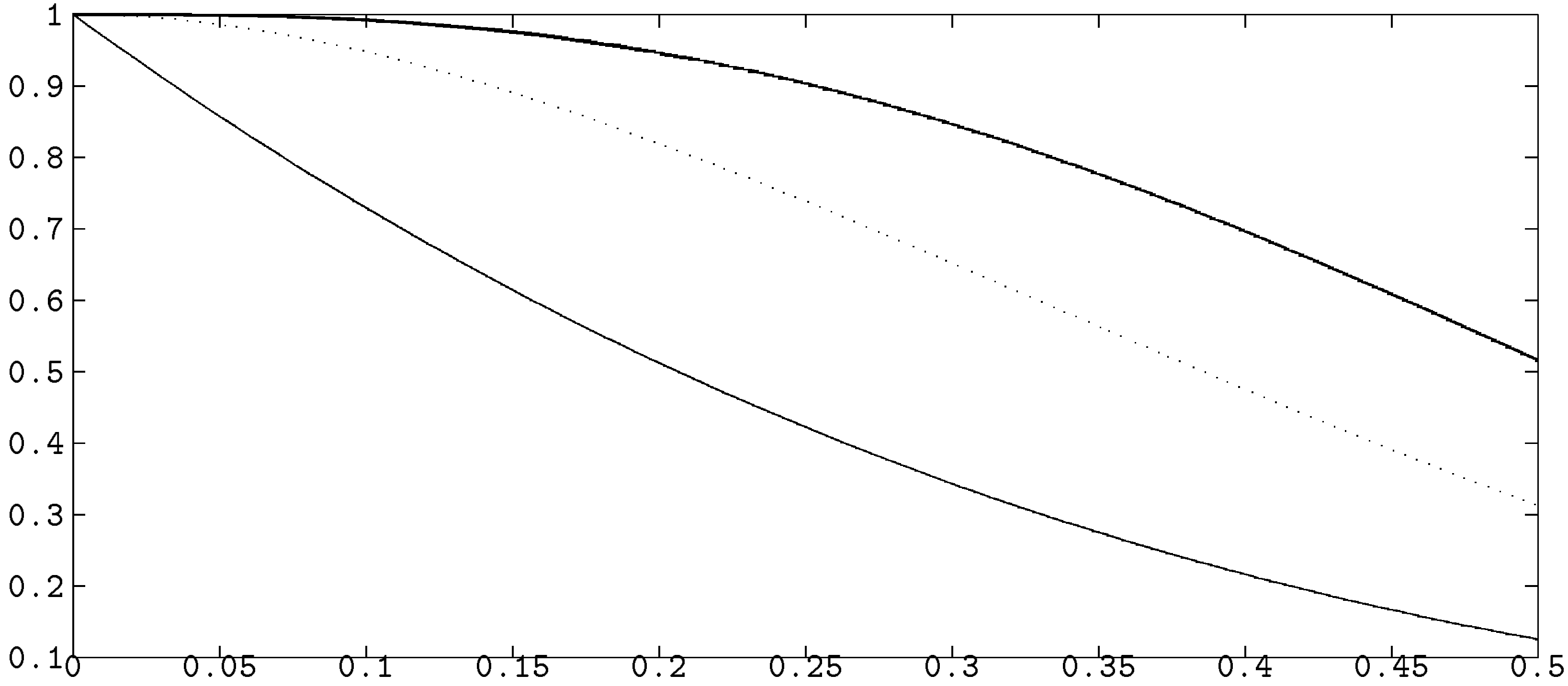
- Connection Establish
  - 3 packet exchange: SYN, SYN-ACK, ACK
  - negociate initial sequence numbers and options
  - data can be sent with the last ACK (in principle)
- Established
  - data is exchanged
- Connection Close
  - 3 packet exchange: FIN, FIN-ACK, ACK
  - improper close using RST becoming common
  - FIN often sent with last data packet

Loss during Connection Establish

- SYN retransmit
  - timeout at 6 seconds, 24 seconds, 48 seconds
  - must establish in 75 seconds or fail
- SYN-ACK
  - this timer can also fire (same intervals)
  - most useful if the last ACK is lost
- Chances of getting connected:
  - assume loss rate is "x"
  - 1st attempt: $(1 - x)^3$
  - within 6 seconds: 1 drop (any of the 3)
  - at all: generally max of 2 drops
  - odd case: > 2 drops if last ACK is dropped

Probability of succesful open with extreme loss

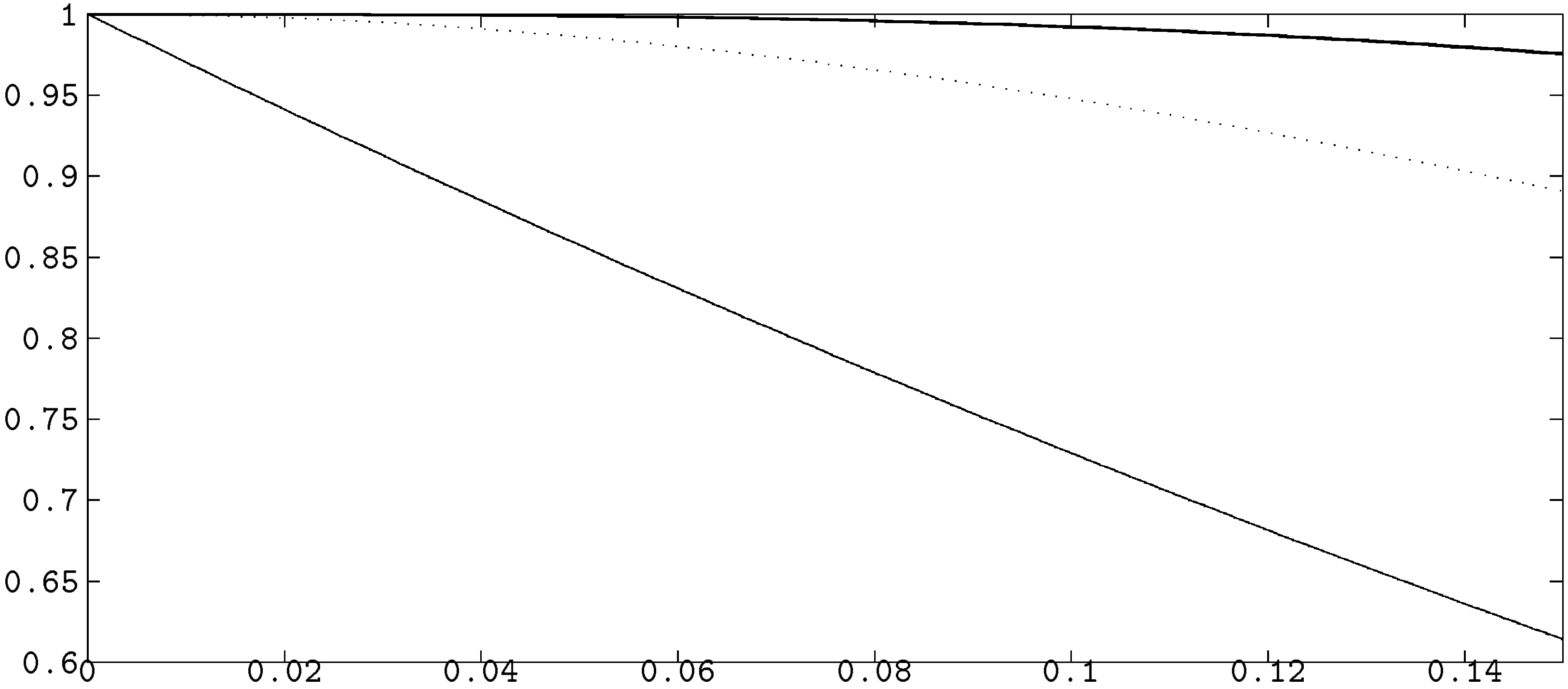
immediate connection, 6 second delay, any connection



At high to extreme loss probability of delay in getting established is quite high. At 15% loss chances of failed connection is 2-3%. At 25% loss chances of failed connection is about 10%. Service will be painfully slow and somewhat unreliable.

Probability of succesful open with high loss

immediate connection, 6 second delay, any connection



Below 5% loss chances of failed connection is near zero and probability of more than 6 second in delay in getting connected is low (under 2%).

Probability of succesful open - summary

- At high to extreme loss probability of delay in getting established is quite high. At 15% loss chances of failed connection is 2-3%. At 25% loss chances of failed connection is about 10%. Service will be painfully slow and somewhat unreliable.

- Below 5% loss chances of failed connection is near zero and probability of more than 6 second in delay in getting connected is low (under 2%).

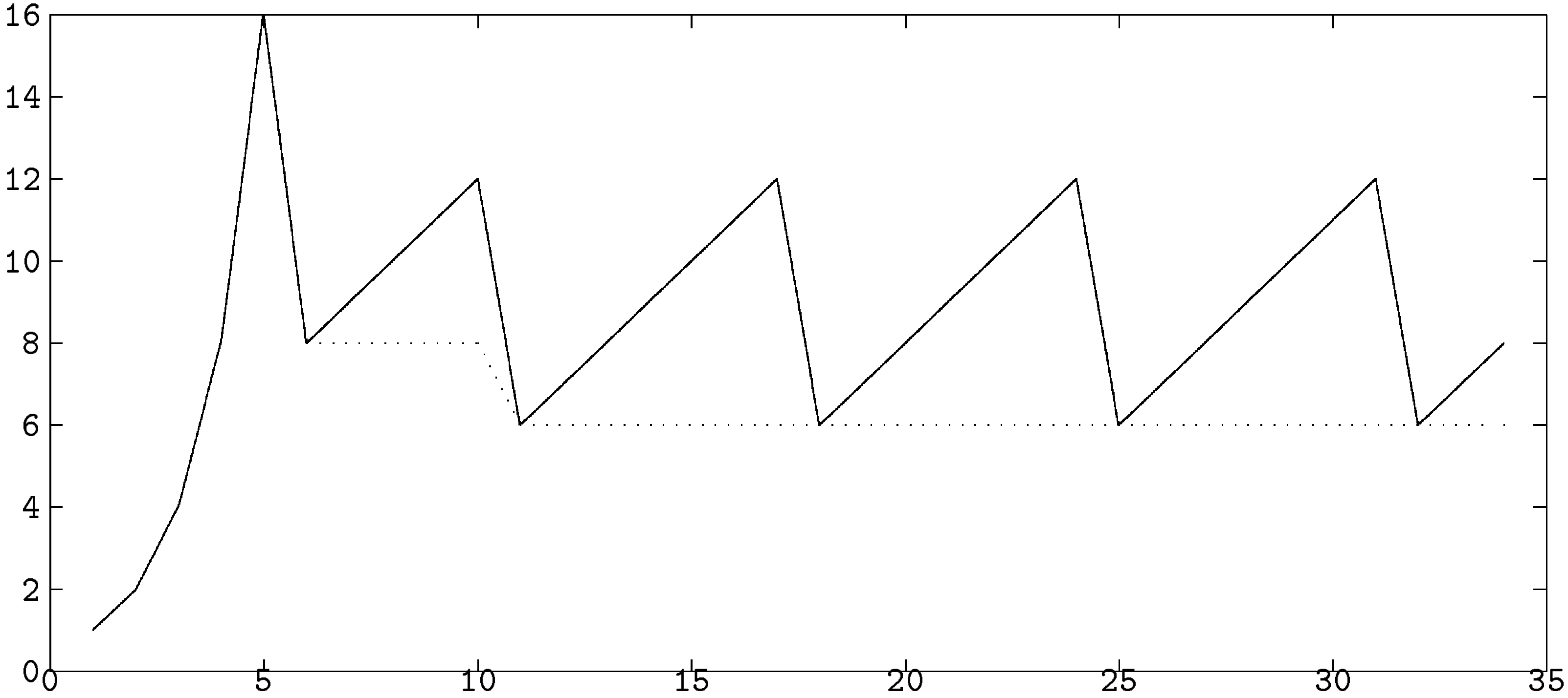
- Caching architecture "first packet" problems may reduce probability of immediate connection, and increase rate of failed connections. May not be an issue for large aggregates or heavily used prefixes. First packet on return path may also be an issue.

Stages of Data Transfer

Initial slow start begins with RTO = 6 seconds, a window of 1 and a congestion avoidance threshhold set to the ceiling. As the window grows exponentially a few things can happen:

- received duplicate ACKs due to an isolated drop
  - fast retransmit
  - fast recovery
  - enter congestion avoidance
- initial timeout
  - due to early or excessive drop or fast retransmit not implemented
  - repeat slow start
  - enter congestion avoidance at 1/2 previous window
- subsequent timeout
  - due to retransmit dropped
  - repeat slow start
  - half the congestion avoidance threshhold again
- transfer completed

TCP Window Size vs Time - Ideal



During initial slow start the window will overshoot the queue capacity and then packet drop will occur. Congestion avoidance then comes into play. Fast retransmit detects the loss more quickly and helps avoid a slow start. Fast recovery keeps data flowing during the one RTT after a packet loss.

TCP Bandwidth – the Golden Rule

No matter how fast the network, the acheived bandwidth is limited by the effective window size divided by the packet round trip time (RTT). This is true for any protocol that sends a limited amount of data in a unacknowledged window.

window	RTT	bandwidth	window	RTT	bandwidth
4 KB	100 usec	327 mb/s	64 KB	70 msec	7.5 mb/s
4 KB	70 msec	468 kb/s	64 KB	250 msec	2.1 mb/s
4 KB	250 msec	131 kb/s	256 KB	70 msec	30 mb/s
8 KB	70 msec	936 kb/s	256 KB	250 msec	8.4 mb/s
8 KB	250 msec	262 kb/s	1 MB	70 msec	120 mb/s
16 KB	70 msec	1.87 mb/s	8 MB	70 msec	960 mb/s
16 KB	250 msec	524 kb/s	32 MB	70 msec	3.8 gb/s

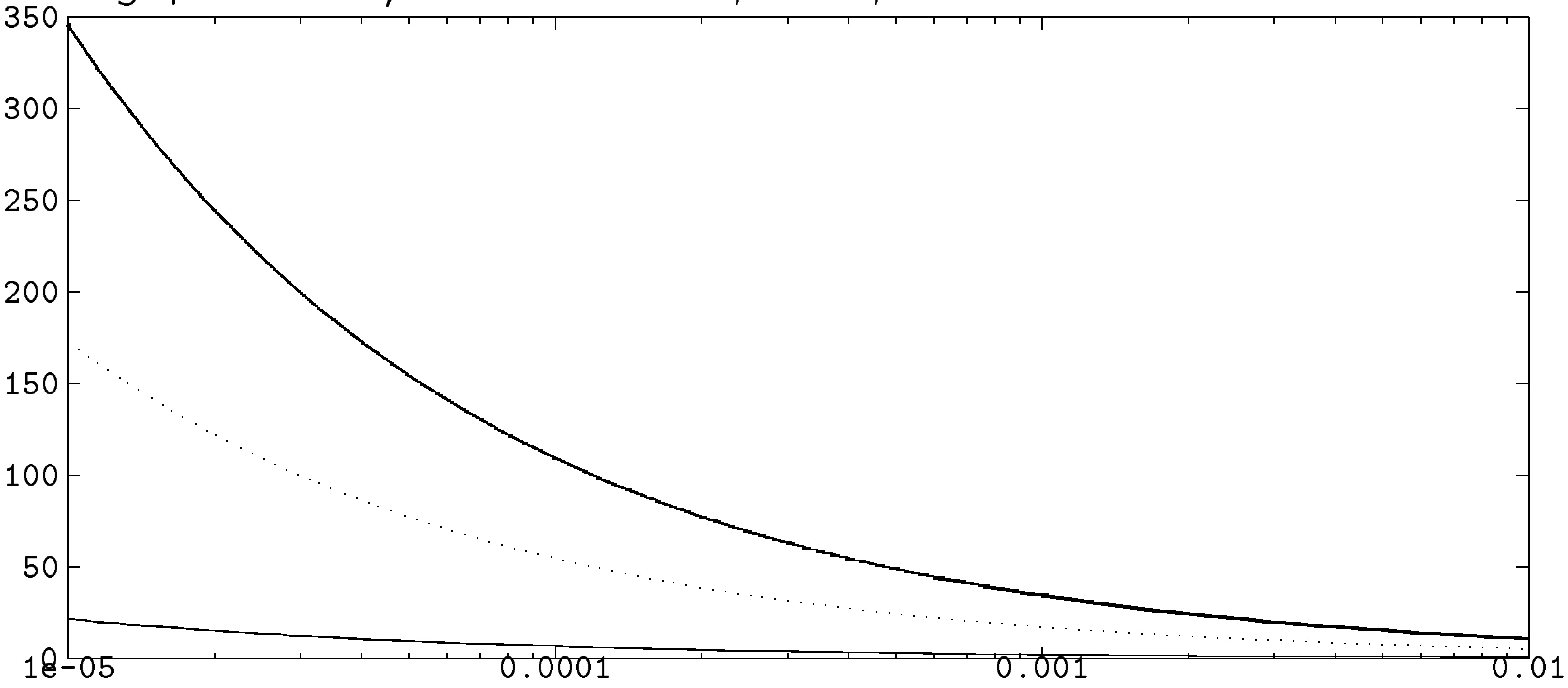
TCP Bandwidth vs Loss Rate - Rough Estimate

Ideally the window will oscillate in size between some maximum value and one half that value. The window increases in size by one TCP segment per RTT, so segment size comes into play. There is exactly one packet loss per cycle.

Average of sawtooth	$W_{avg} = \frac{1}{2}(W_{max} + \frac{W_{max}}{2})$
"golden rule"	$BW = \frac{W_{avg}}{RTT}$
one loss per cycle	$\text{pps loss} = \frac{1}{\frac{\frac{1}{2} W_{max}}{segsize} RTT}$
the data rate in pps	$\text{pps data} = \frac{BW}{segsize}$
the loss rate	% loss = pps loss / pps data
substitute	$\%\ \text{loss} = \frac{2}{3} (\frac{segsize}{BW\ RTT})^2$
solve for BW	$BW^2 = \frac{3}{2} \frac{segsize^2}{RTT^2\ loss}$

TCP Bandwidth vs Loss – Rough Plot – High Speed

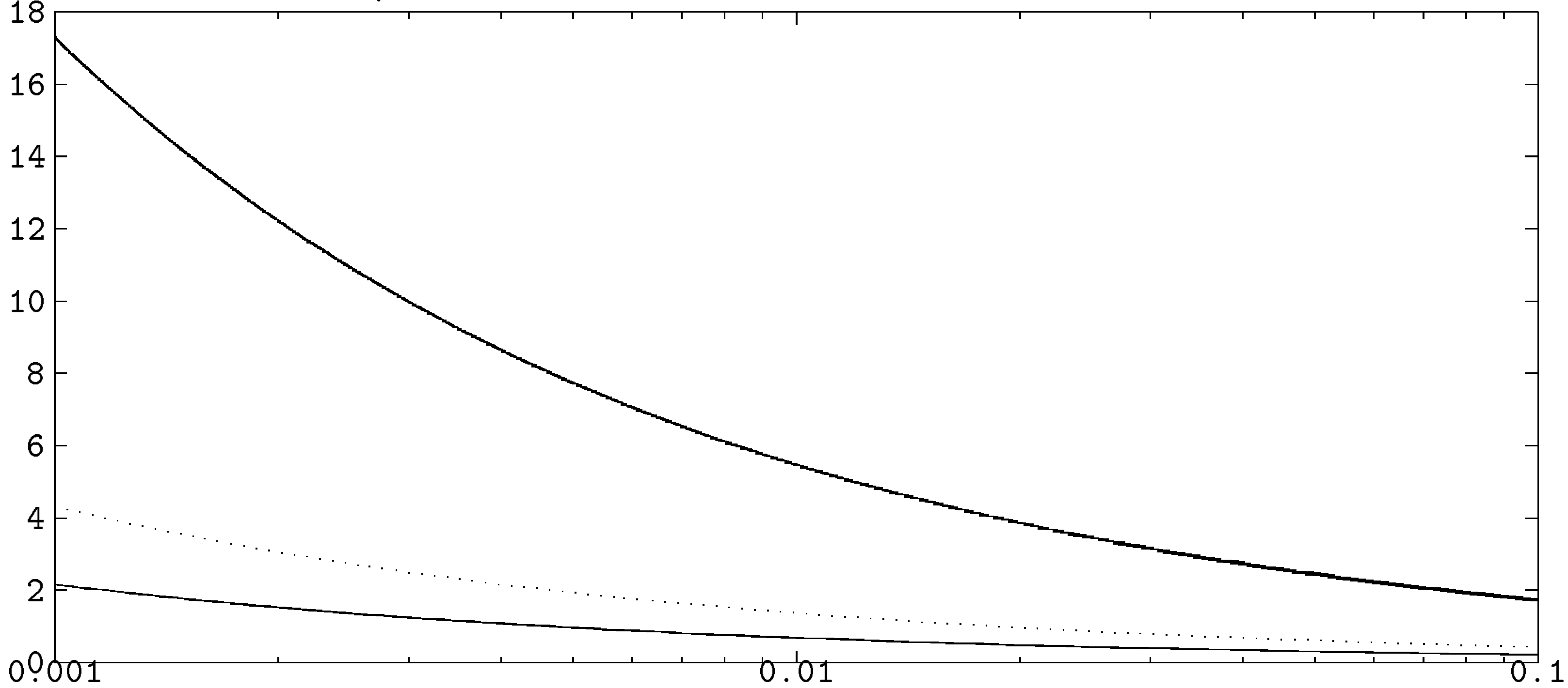
throughput in mb/s vs loss – 512, 4096, and 8192 MSS – 70 msec RTT



Extremely low loss and large TCP segments are needed to sustain high delay bandwidth product TCP flows

TCP Bandwidth vs Loss Rate - Rough Plot – Low Speed

throughput in mb/s vs loss – 512, 1024, and 4096 MSS – 70 msec RTT



Single TCP flows of even tens of kb/s require fairly low loss. The plot actually represents an upper bound on the bandwidth that can be acheived with a given amount of loss. The plot goes to 5% loss, though the model becomes highly inaccurate for small TCP window sizes.

Accuracy of Bandwidth vs Loss Estimate

- The estimate represents an upper bound on BW
- Also a upper bound on loss needed constrain BW
- accuracy is poor at smaller window sizes
- loss is unlikely to be evenly spaced
- at smaller window sizes chances of missing fast retransmit increases

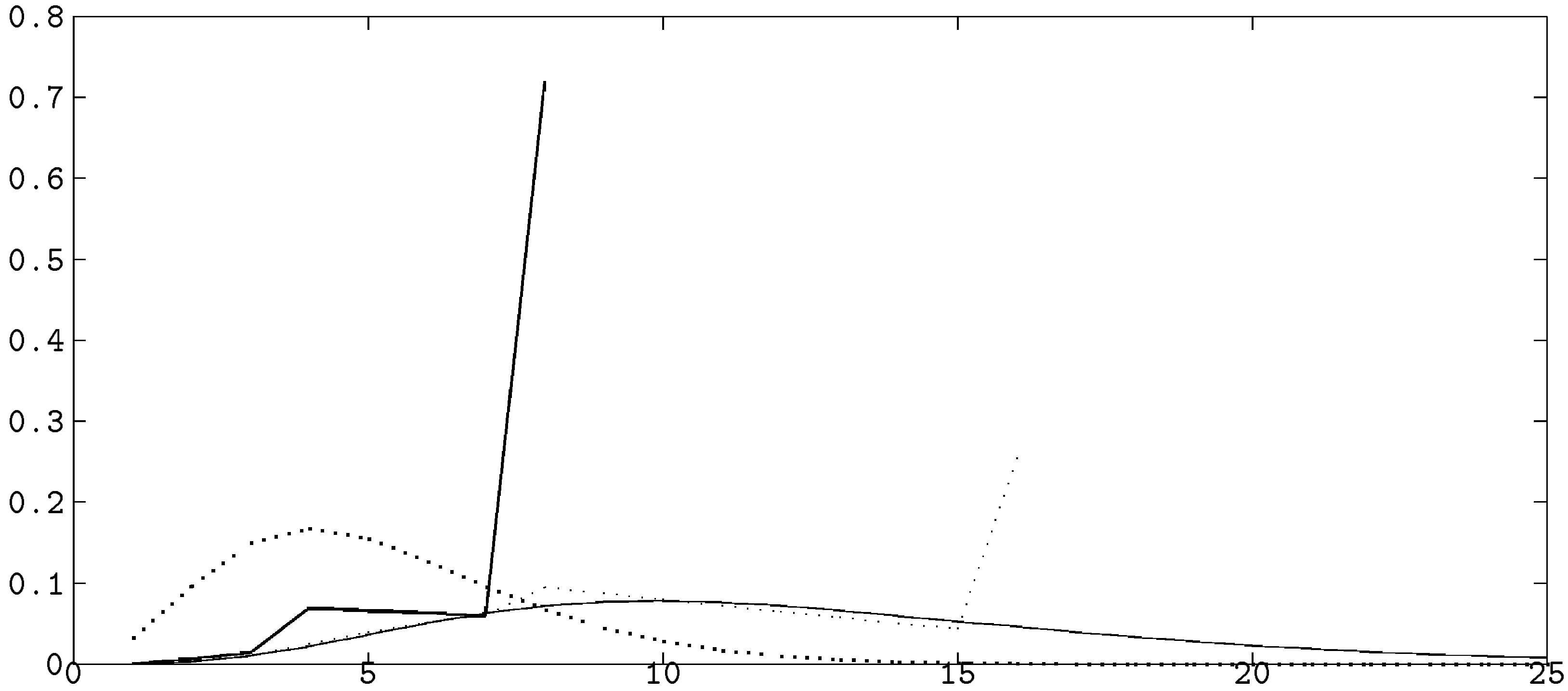
Probability of Various Window Sizes

Define "P(n)" to be the probability of window size "n" with TCP in steady state. Define "x" to be the loss rate.

$$P(1) = x\ P(1) + (1-(1-x)^2)\ P(2) + (1-(1-x)^3)\ P(3)$$
$$P(2) = (1-x)\ P(1) + (1-(1-x)^4)\ P(4)$$
$$P(n) = (1-x)\ P(n-1) + (1-(1-x)^{2n})\ P(2n)$$
$$1 = \sum_{n=1}^{\infty} P(n)$$

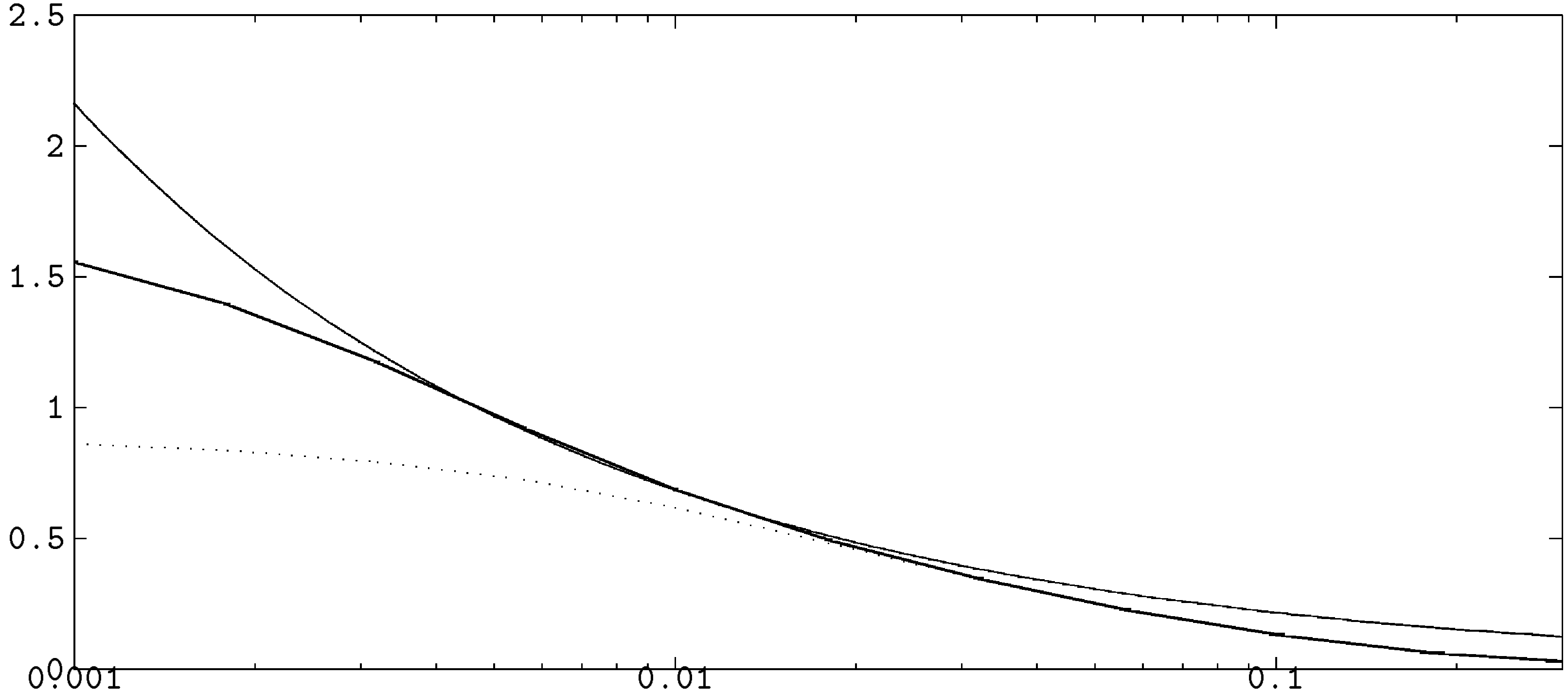
- Windows below size of 4 will fail to get 3 duplicate ACKs on a loss, yielding the special case for P(1).
- Neglects effects of multiple drop within a window
- Assume P(n) is 0 above some value of "n".
- Fill in a matrix and solve equations
- Estimate average window size and bandwidth

Window Size Probability Distributions



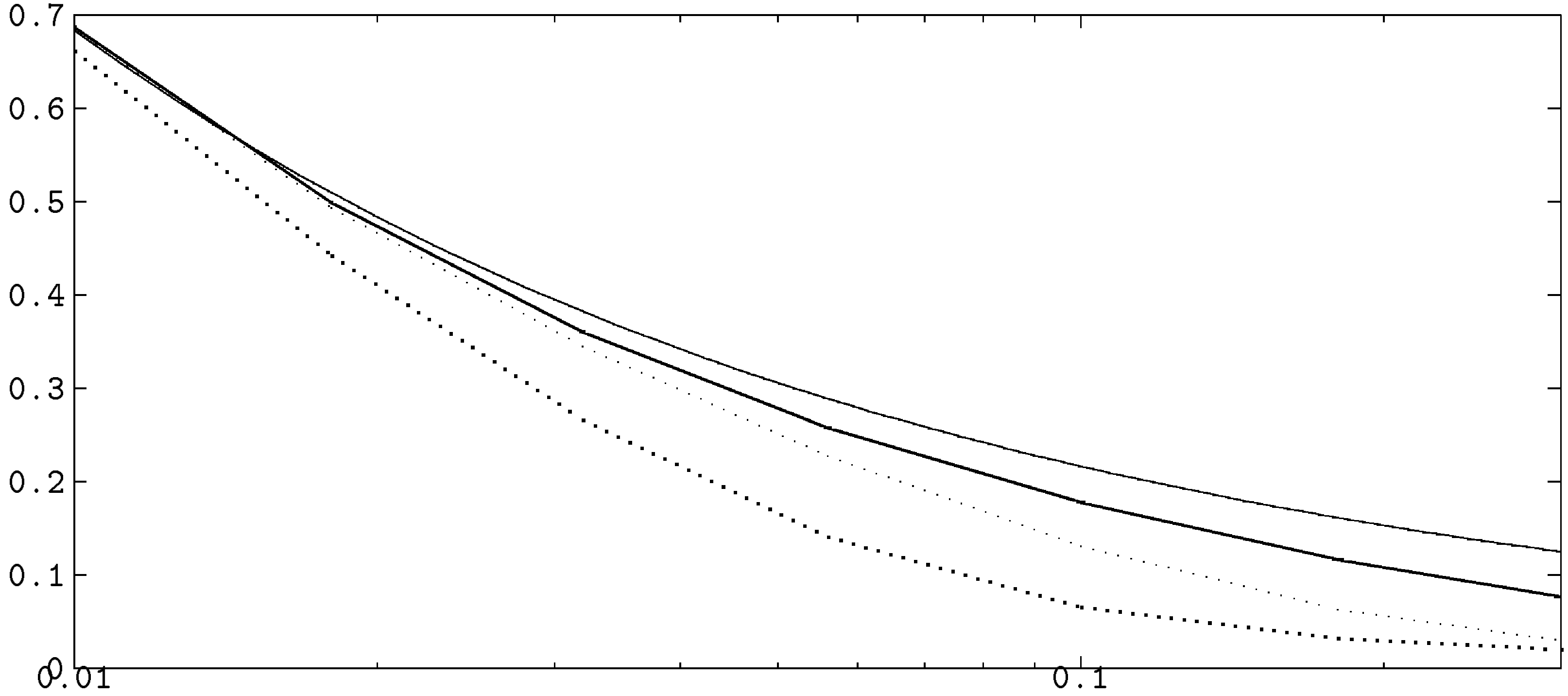
- 512 byte MSS, 70 msec RTT
- 1 plot is 5% loss with a 4KB window limit
- 3 plots are 1% loss with 4KB, 8KB, 16KB window limit
- shows the effect of constrained window

Comparing Methods of Predicting TCP Performance



- 512 byte MSS, 70 msec RTT
- top plot is previously derived upper bound
- next plot is window contrained to 16KB
- bottom plot is window contrained to 8KB

TCP Performance at Small Window Sizes



- 512 byte MSS, 70 msec RTT, 16KB window limit
- top line is previously derived upper bound
- next line neglects loss of fast retransmit
- next assumes 4*RTT timeout below window of 4
- next assumes 4*RTT timeout below window of 7

TCP Response Under Loss Conditions - Conclusions

- TCP is able to connect reliably up to very high loss
- TCP may take a while to connect at high loss rates
- Very high performance requires very low loss
- Performance drops off above 1% loss
- TCP is able to move bits under high loss conditions
- At high loss rates response may get very slow
- TCP dominated networks are *NOT* prone to congestion collapse but are more efficient with a smaller number of flows with larger windows than with lots of smaller flows