The UNIVERSITY *of* NORTH CAROLINA
at CHAPEL HILL

The Effects of Active Queue Management on Web Performance

The Good, the Bad, and the Ugly

Long Le, Jay Aikat, Kevin Jeffay, and Don Smith

November 2003

http://www.cs.unc.edu/Research/dirt

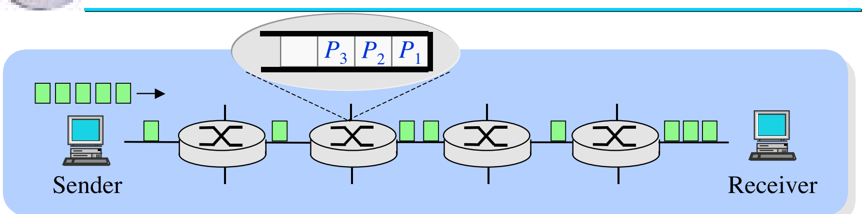
The Effects of AQM on the Web

Outline

- Review: Congestion control on the Internet today
- Router-based congestion control
  - Active Queue Management
  - Explicit Congestion Notification
- State of the art in active queue management (AQM)
  - Control theoretic *v*. traditional randomized dropping AQM
- Do AQM schemes work?
  - An empirical study of the effect of AQM on web performance
- Analysis of AQM performance
  - The good, the bad, and the ugly…

Congestion Control on the Internet

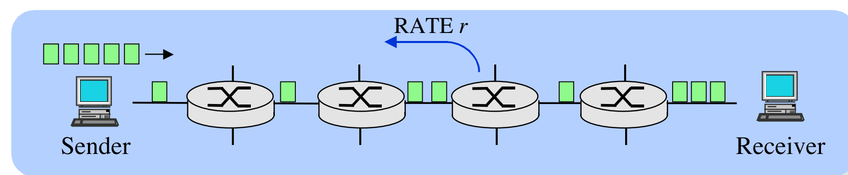
The end-to-end approach



- Congestion control is the problem of ensuring queues at switches in the network don't fill to capacity
- Operationally, congestion control is the problem of determining how fast to transmit data
  - When can an end-system speed up?
  - When should it slow down?

Congestion Control on the Internet

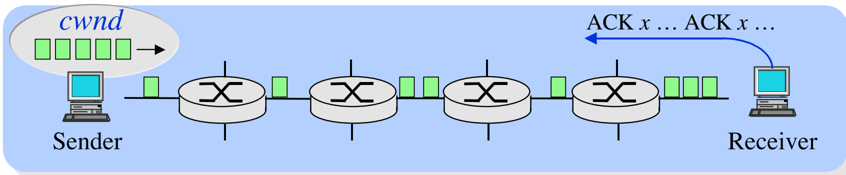
The end-to-end approach



- The Internet was founded on the principle of end-to-end control
  - End-systems must determine on their own if the network is congested
  - Congestion is inferred by observing loss and/or delay
- (Alternative: Hop-by-hop congestion control:
  - Switches provide congestion feedback to end-systems)

Congestion Control on the Internet

The end-to-end approach



- TCP's congestion control algorithm:
  - Sender maintains a variable-sized buffer of packets to be transmitted (called the "congestion window"— *cwnd*)
  - The congestion window represents the maximum amount of data a connection can have outstanding (unacknowledged) in the network
  - The congestion window grows as ACKs are received at the sender

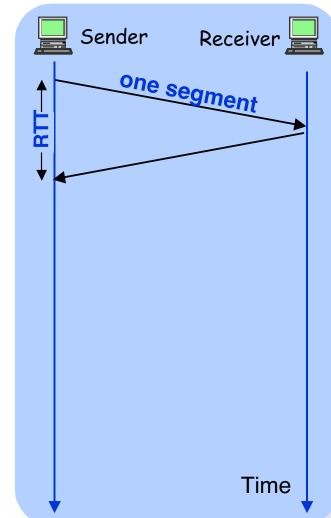
TCP Congestion Control

Congestion window evolution

- A connection's transmission rate is:

$$rate = \frac{cwnd \cdot segment\ size}{RTT}$$

- TCP uses two algorithms (run serially) to set *cwnd*:
  - Slowstart
  - Congestion avoidance



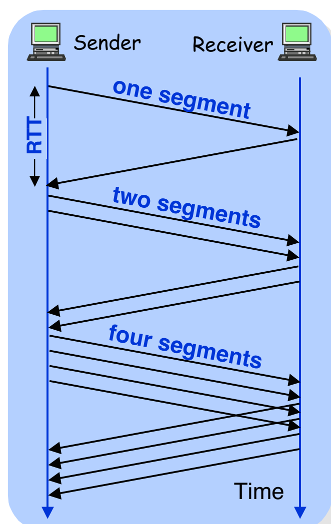
TCP Congestion Control

Congestion window evolution

- A connection's transmission rate is:

$$rate = \frac{cwnd \cdot segment\ size}{RTT}$$

- *Slowstart*: Sender increases its congestion window by 1 segment for each ACK (*i.e.*, 1 segment each RTT)
  - Exponential increase in window size each RTT
  - ("Slowstart" not so slow!)



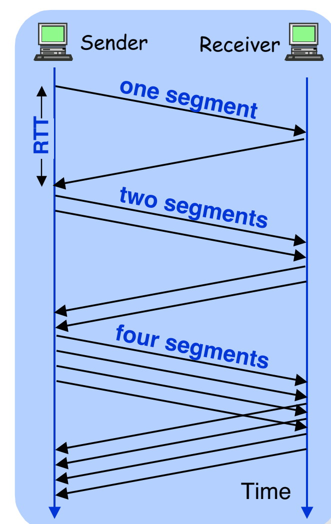
TCP Congestion Control

Congestion window evolution

- A connection's transmission rate is:

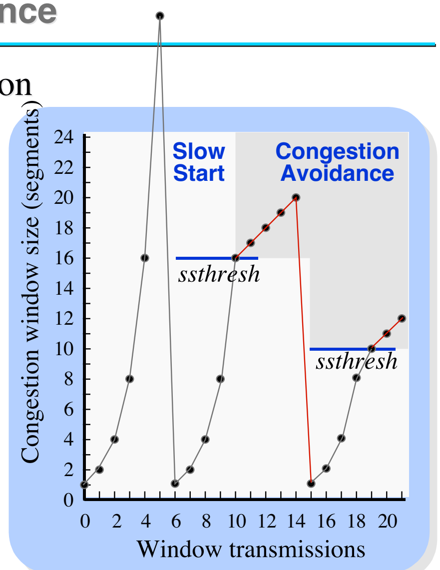
$$rate = \frac{cwnd \cdot segment\ size}{RTT}$$

- *Slowstart*: Sender increases its congestion window by 1 segment each RTT until:
  - Loss occurs
  - *cwnd == ssthresh threshold*
- When slowstart threshold is reached TCP connection enters *congestion avoidance* state



TCP Congestion Control
Congestion avoidance

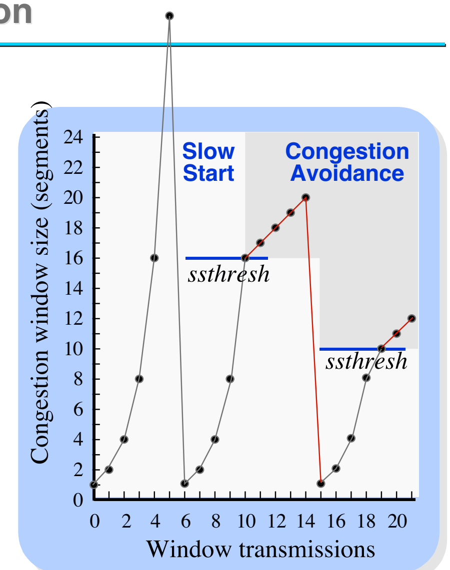
- Sender increases its congestion window by 1 segment each *cwnd* transmissions until
  - Loss occurs, or
  - Maximum window size is reached
- When loss occurs
  - slowstart threshold *ssthresh* is set to 1/2 *cwnd*
  - *cwnd* is set to 1 segment, and
  - slowstart is reentered



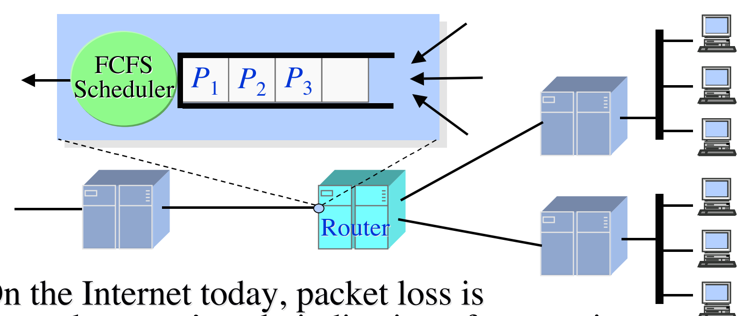
TCP Congestion Control
AIMD rate adaptation

- Arithmetic increase:
  - Increase by 1 segment per *cwnd* during congestion avoidance
  - Linear probing for available bandwidth
- Multiplicative decrease:
  - Decrease threshold by 1/2 when loss occurs
  - React quickly when the network is congested

$$rate = \frac{cwnd \cdot segment\ size}{RTT}$$



Congestion Control on the Internet
Summary



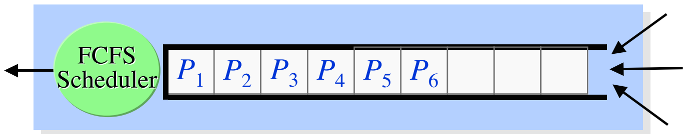
- On the Internet today, packet loss is the end-system's only indication of congestion
- As switch's queues overflow, arriving packets are dropped
  - "Drop-tail" FIFO queuing is the default
- TCP end-systems detect loss and respond by reducing their transmission rate

The Effects of AQM on the Web
Outline

- Review: Congestion control on the Internet today
- Router-based congestion control
  - Active Queue Management
  - Explicit Congestion Notification
- State of the art in active queue management (AQM)
  - Control theoretic *v.* traditional randomized dropping AQM
- Do AQM schemes work?
  - An empirical study of the effect of AQM on web performance
- Analysis of AQM performance
  - The good, the bad, and the ugly…

The Case Against Drop-Tail

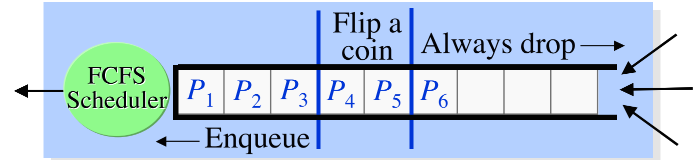
Towards router-based congestion control



- Large (full) queues in routers are a bad thing
  - End-to-end latency is dominated by the length of queues at switches in the network
- Allowing queues to overflow is a bad thing
  - Connections that transmit at high rates can starve connections that transmit at low rates
  - Causes connections to synchronize their response to congestion and become unnecessarily bursty

Router-Based Congestion Control

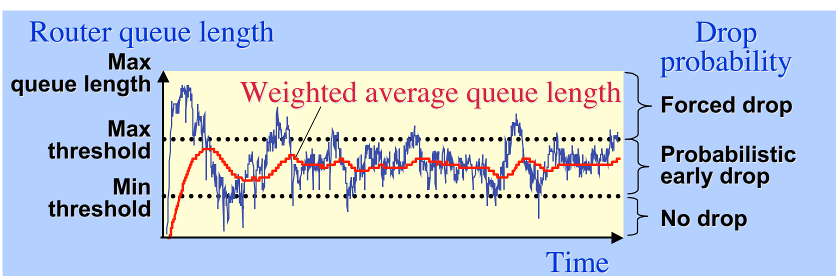
Active queue management (AQM)



- Key concept: Drop packets *before* a queue overflows to signal *incipient* congestion to end-systems
- Basic mechanism: When the queue length exceeds a threshold, packets are probabilistically dropped
- *Random Early Detection* (RED) AQM:
  - Always enqueue if queue length less than a low-water mark
  - Always drop if queue length is greater than a high-water mark
  - Probabilistically drop/enqueue if queue length is in between

Active Queue Management

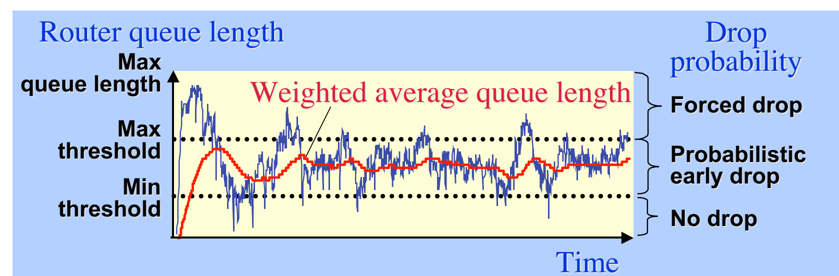
The RED Algorithm [Floyd & Jacobson 93]

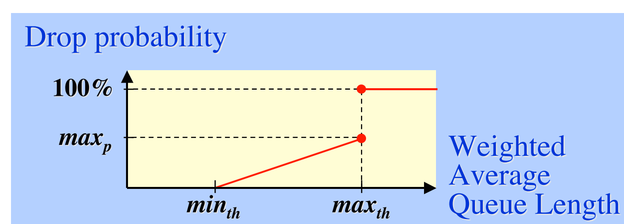


- RED computes a weighted moving average of queue length to accommodate bursty arrivals
- Drop probability is a function of the current average queue length
  - The larger the queue, the higher the drop probability

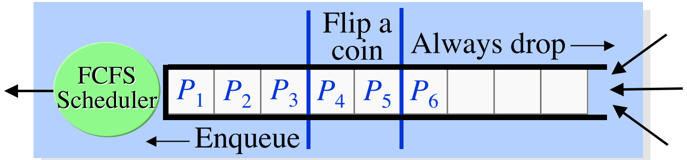
Active Queue Management

The RED Algorithm [Floyd & Jacobson 93]



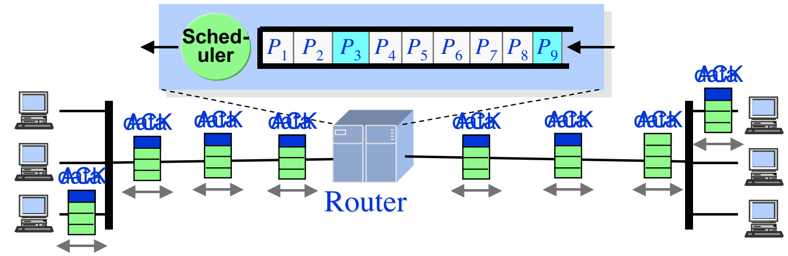


Active Queue Management
Explicit Congestion Notification (ECN)



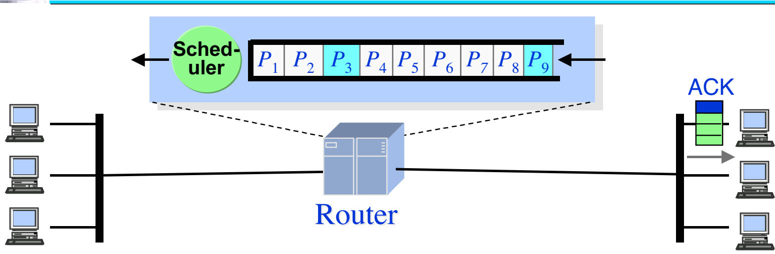
- Dropping packets is a simple means of signaling congestion but it's less than ideal
  - It may take a long time for a sender to detect and react to loss, hence congestion signaled by packet drops may be ineffective
  - There are subtle fairness issues in the way flows are treated
- ECN: Instead of dropping packets, send an explicit signal back to the sender to indicate congestion
  - (An old concept: ICMP Source Quench, DECbit, ATM, …)

Explicit Congestion Notification
Overview



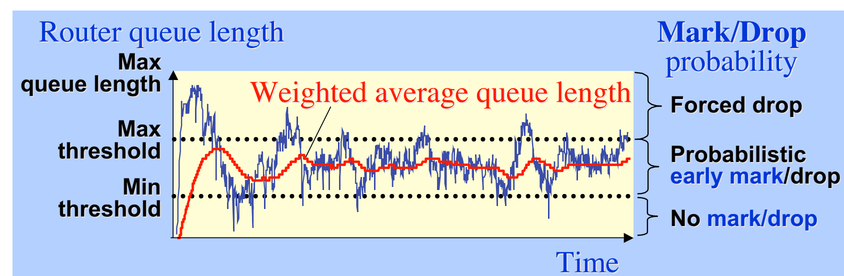
- Modify a RED router to "mark" packets rather than dropping them
- Set a bit in a packet's header and forward towards the ultimate destination
- A receiver recognizes the marked packet and sets a corresponding bit in the next outgoing ACK

Explicit Congestion Notification
Overview



- When a sender receives an ACK with ECN it invokes a response similar to that for packet loss:
  - Reset the congestion window *cwnd* and halve the slow-start threshold *ssthresh*
  - Continue to use ACK-clocking to pace transmission of data packets

Explicit Congestion Notification
Putting the pieces together: AQM + ECN



- If a RED router detects congestion it will mark arriving packets
- The router will then forward marked packets from ECN-capable senders…
- …and drop marked packets from all other senders

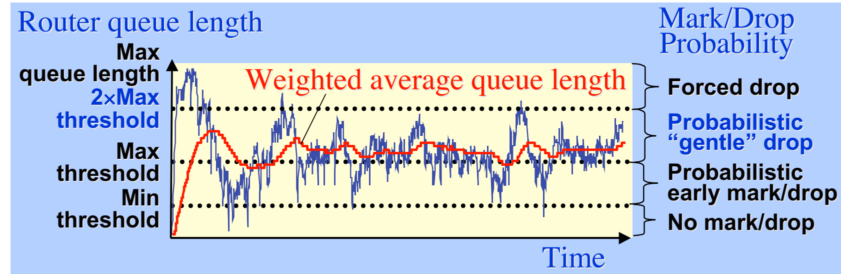
The Effects of AQM on the Web

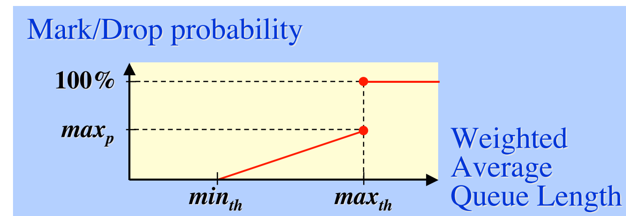
Outline

- Review: Congestion control on the Internet today
- Router-based congestion control
  - Active Queue Management
  - Explicit Congestion Notification
- State of the art in active queue management (AQM)
  - Control theoretic *v.* traditional randomized dropping AQM
- Do AQM schemes work?
  - An empirical study of the effect of AQM on web performance
- Analysis of AQM performance
  - The good, the bad, and the ugly…

The State of the Art in AQM

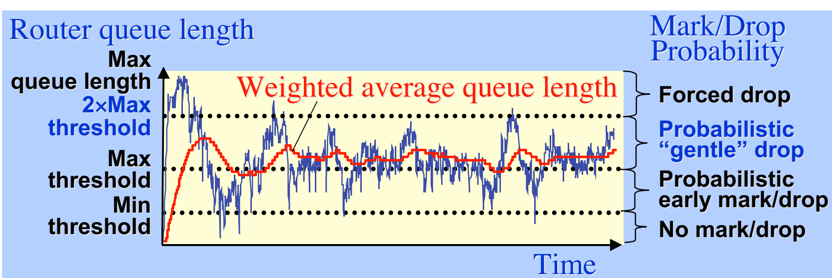
Adaptive/Gentle RED (ARED)

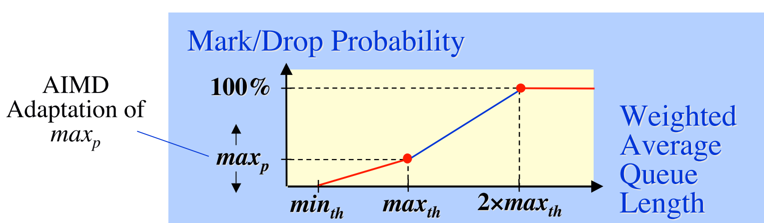




The State of the Art in AQM

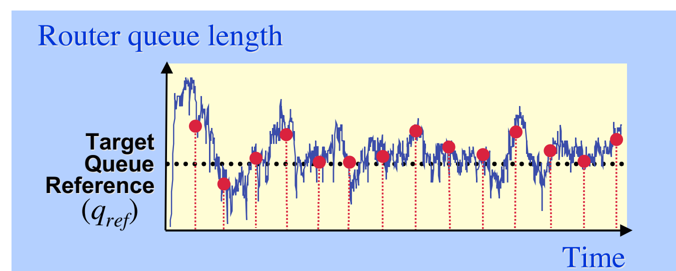
Adaptive/Gentle RED (ARED)





The State of the Art in AQM

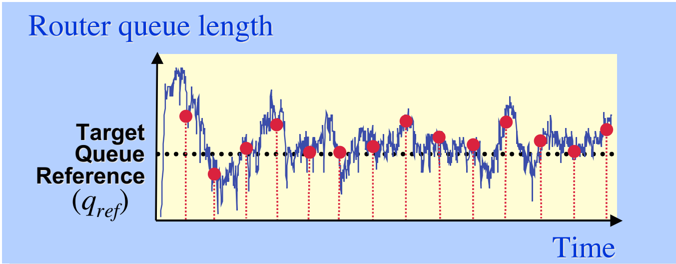
The Proportional Integral (PI) controller



- PI attempts to maintain an explicit target queue length
- PI samples instantaneous queue length at fixed intervals and computes a mark/drop probability at k^{th} sample:
  - $p(kT) = a \times (q(kT) - q_{ref}) - b \times (q((k\text{-}1)T) - q_{ref}) + p((k\text{-}1)T)$
  - a, b, and T depend on link capacity, maximum RTT and the number of flows at a router

The State of the Art in AQM

Random Exponential Marking (REM)



- REM is similar to PI (though differs in details)
- REM mark/drop probability depends on:
  - Difference between input and output rate
  - Difference between instantaneous queue length and target
  - $p(t) = p(t-1) + \gamma\,[\alpha\,(q(t) - q_{ref}) + x(t) - c]$
  - $prob(t) = 1 - \phi^{-p(t)}, \ \phi > 1$ a constant

Do AQM Schemes Work?

(Why do we care?)

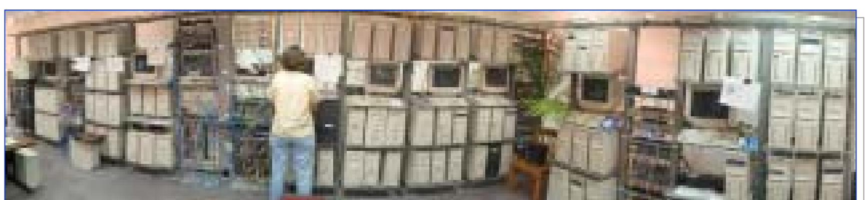
- RFC 2309 strongly advocates deployment of RED active queue management in routers:

  "All available empirical evidence shows that the deployment of active queue management mechanisms in the Internet would have substantial performance benefits. There are seemingly no disadvantages to using the RED algorithm, and numerous advantages. Consequently, we believe that RED active queue management algorithm should be widely deployed."

- Why do we care about the effect of AQM on Web traffic?
  - Web traffic makes up a significant fraction of traffic on most links
  - In theory, a key goal of AQM is to "provide lower delays for interactive applications such as web browsing"

Do AQM Schemes Work?

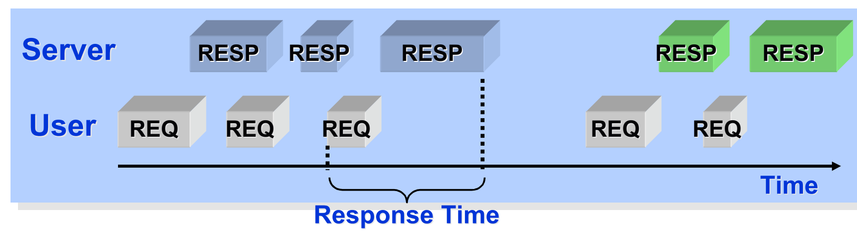
Evaluation methodology



- Evaluate AQM schemes through "live simulation"
- Emulate the browsing behavior of a large population of users surfing the web in a laboratory testbed
  - Construct a physical network emulating a congested peering link between two ISPs
  - Generate synthetic HTTP requests and responses but transmit over real TCP/IP stacks, network links, and switches

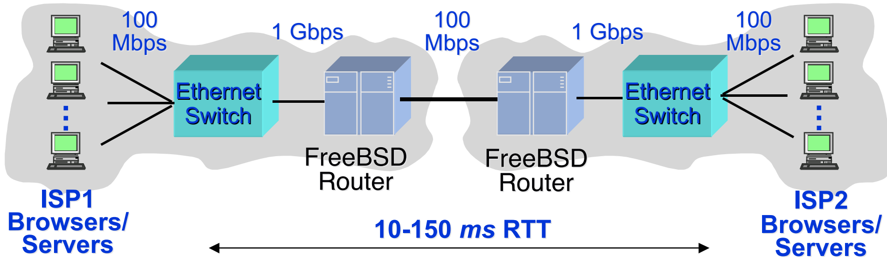
Experimental Methodology

HTTP traffic generation



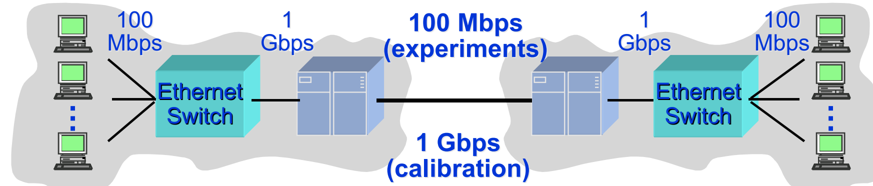
- Synthetic web traffic generated using the UNC HTTP model [SIGMETRICS 2001, MASCOTS 2003]
- Primary random variables:
  - Request sizes/Reply sizes
  - User think time
  - Persistent connection usage
  - Nbr of objects per persistent connection
  - Number of embedded images/page
  - Number of parallel connections
  - Consecutive documents per server
  - Number of servers per page

Experimental Methodology
Testbed emulating an ISP peering link



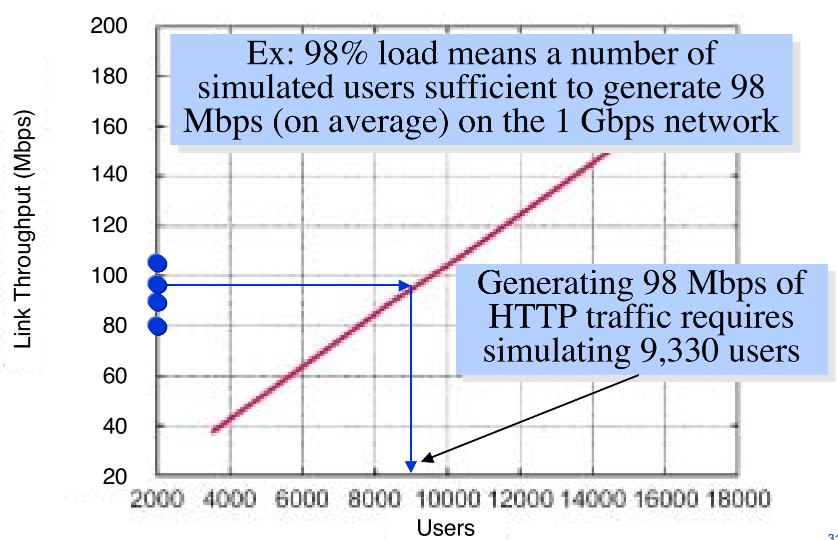
- AQM schemes implemented in FreeBSD routers using ALTQ kernel extensions
- End-systems either a traffic generation client or server
  - Use *dummynet* to provide *per-flow* propagation delays
  - Two-way traffic generated, equal load generated in each direction

Experimental Methodology
1 Gbps network calibration experiments



- Experiments run on a congested 100 Mbps link
- Primary simulation parameter: Number of simulated browsing users
- Run calibration experiments on an uncongested 1 Gbps link to relate simulated user populations to average link utilization
  - (And to ensure offered load is linear in the number of simulated users — *i.e.*, that end-systems are not a bottleneck)

Experimental Methodology
1 Gbps network calibration experiments



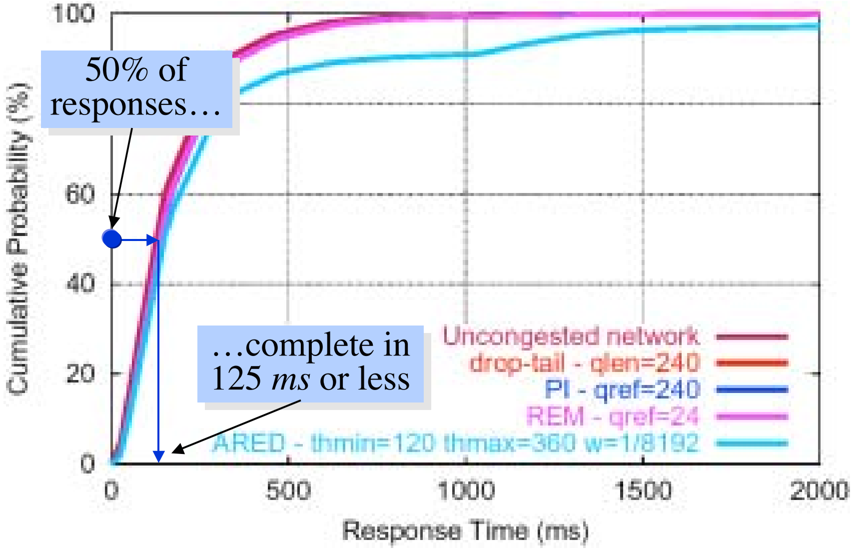
Experimental Methodology
Experimental plan

	80% 90% 98% 105%
uncongested	*loss rate*
drop-tail	*utilization*
ARED	*response times*
PI	*completed requests*
REM	

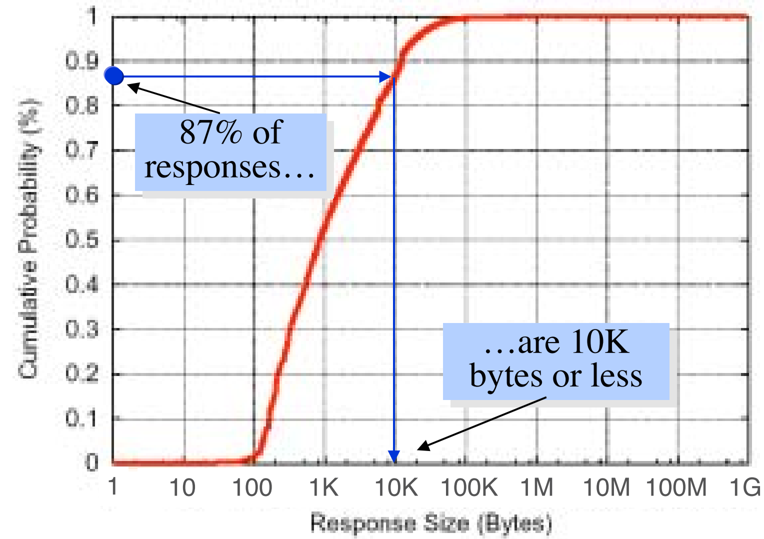
- Run experiments with ARED, PI, and REM using their recommended parameter settings at different offered loads
- Compare results with drop-tail FIFO at the same offered loads…
  - (the "negative" baselines — the performance to beat)

  …and compare with performance on the 1 Gbps network
  - (the "positive" baseline — the performance to achieve)
- Redo the experiments with ECN

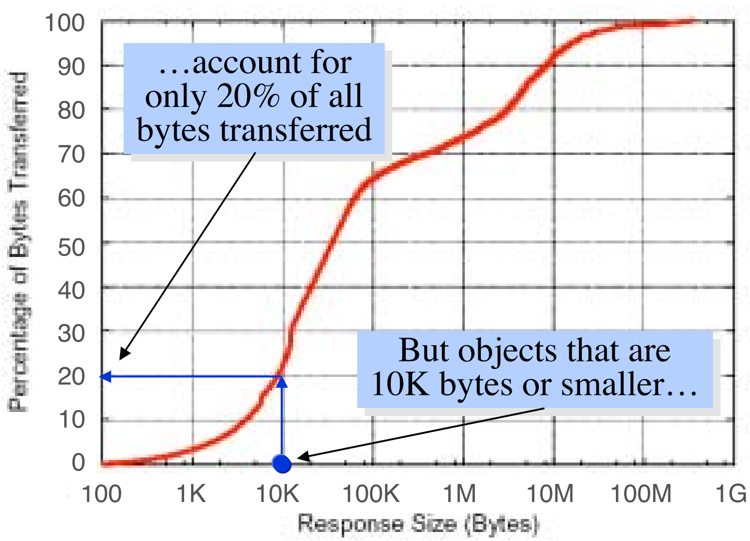
Experimental Results — 80% Load
Performance with packet drops



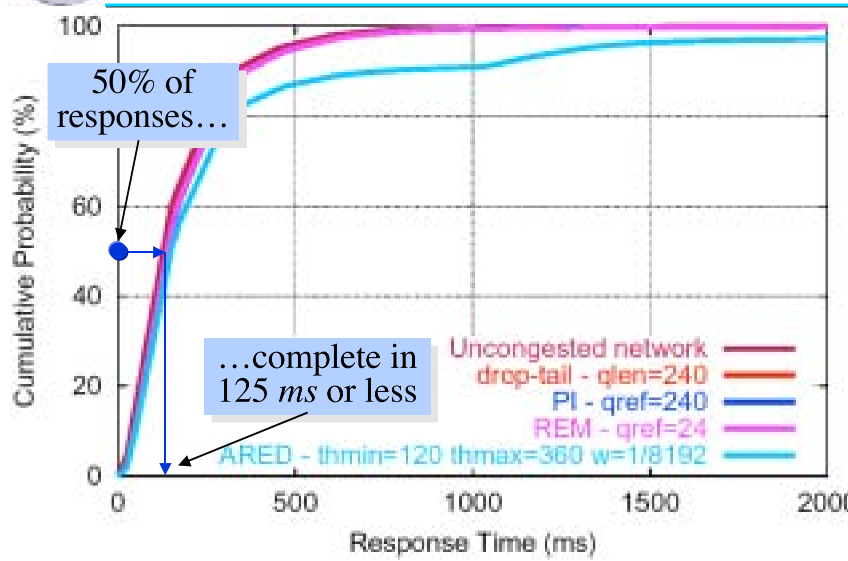
The Structure of Web Traffic
Distribution of response sizes



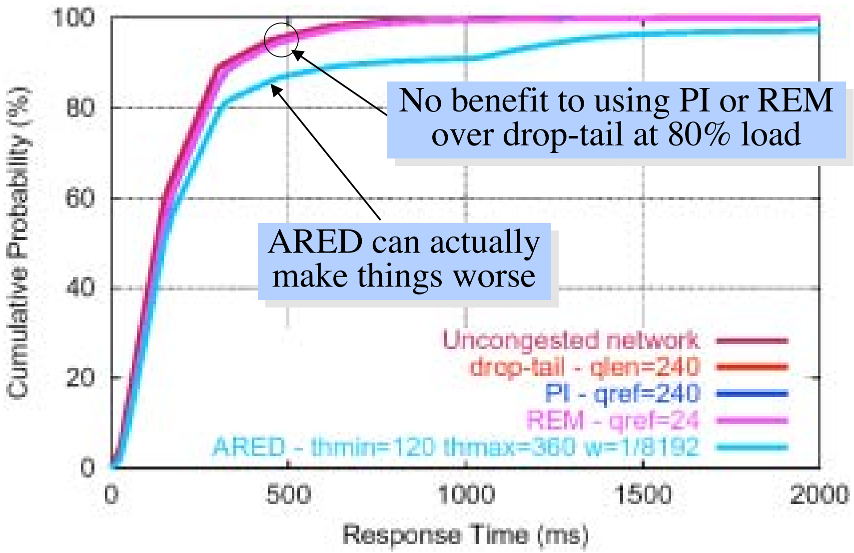
The Structure of Web Traffic
Percent of bytes transferred by response sizes



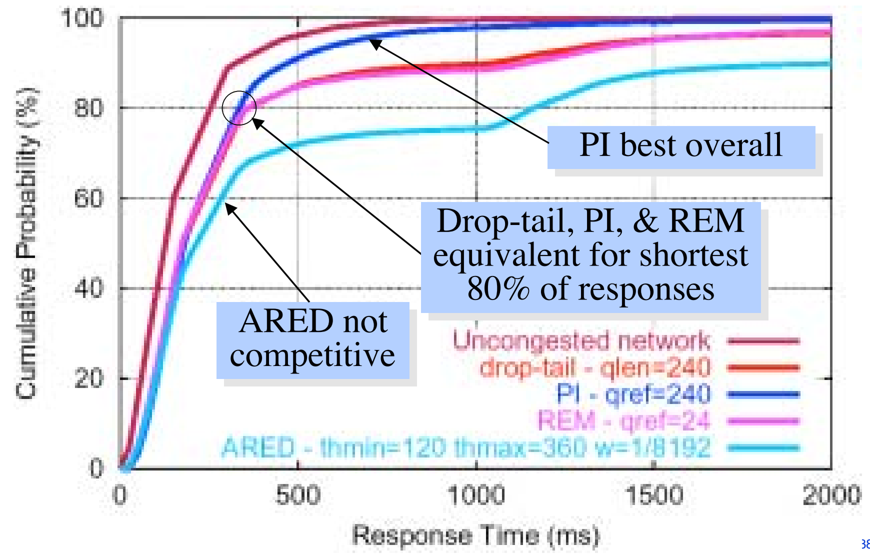
Experimental Results — 80% Load
Performance with packet drops



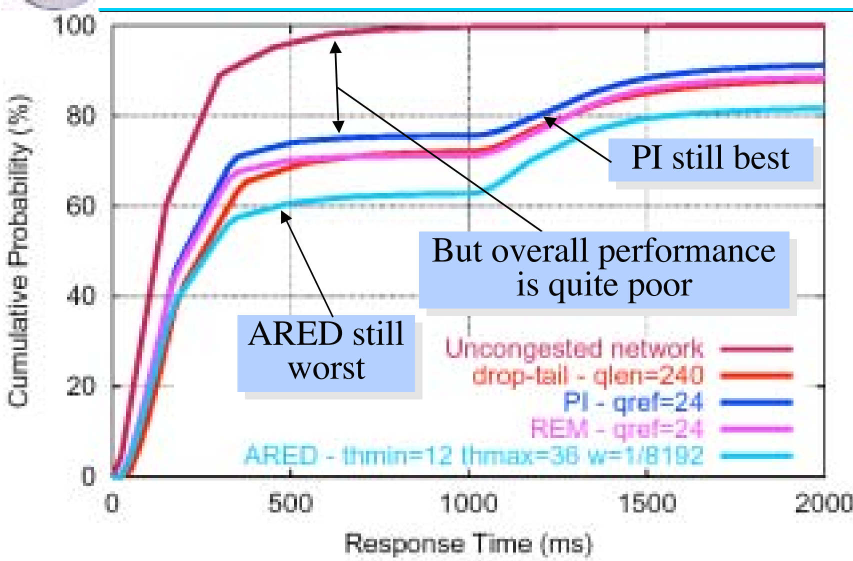
Experimental Results — 80% Load
Performance with packet drops



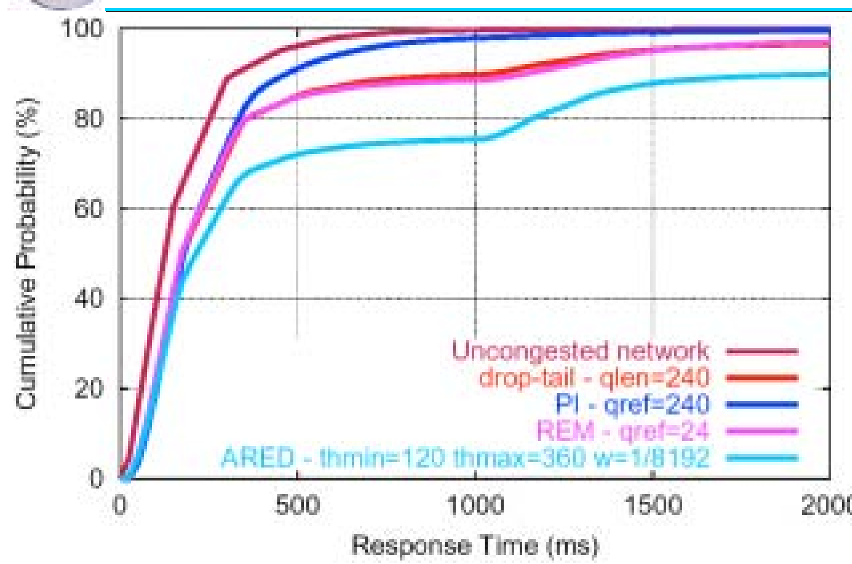
Experimental Results — 90% Load
Performance with packet drops



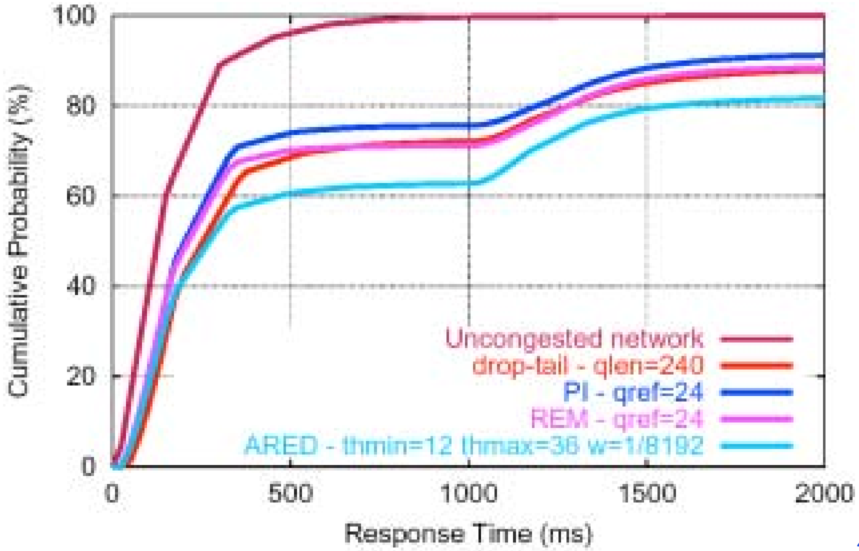
Experimental Results — 98% Load
Performance with packet drops



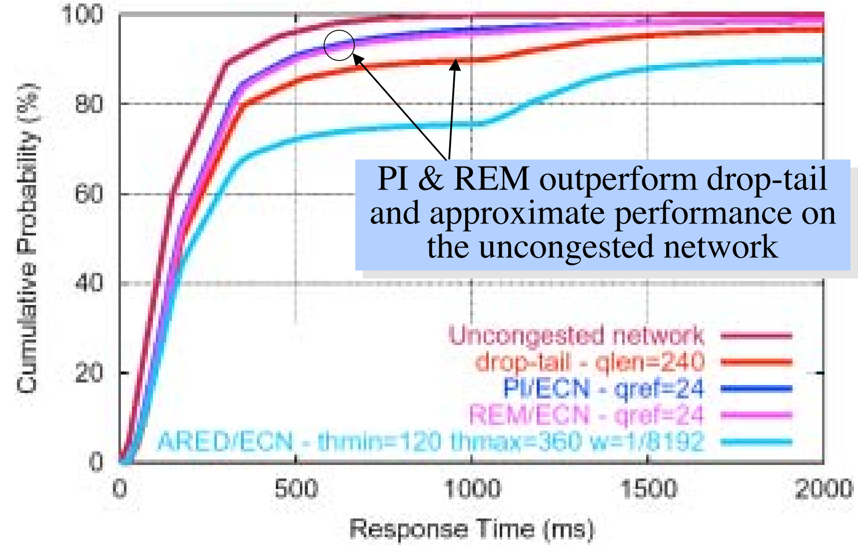
Experimental Results — 90% Load
Performance with packet drops



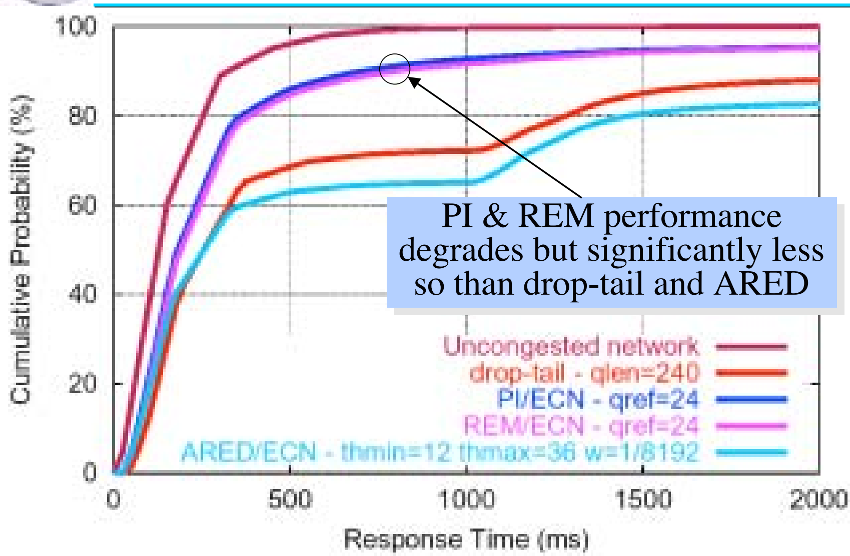
Experimental Results — 98% Load
Performance with packet drops



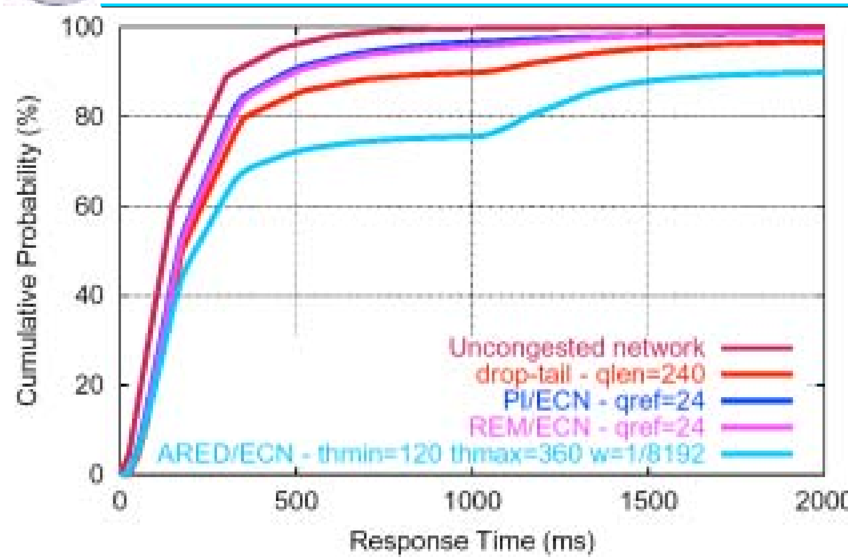
ECN Results — 90% Load
Comparison of all schemes



ECN Results — 98% Load
Comparison of all schemes

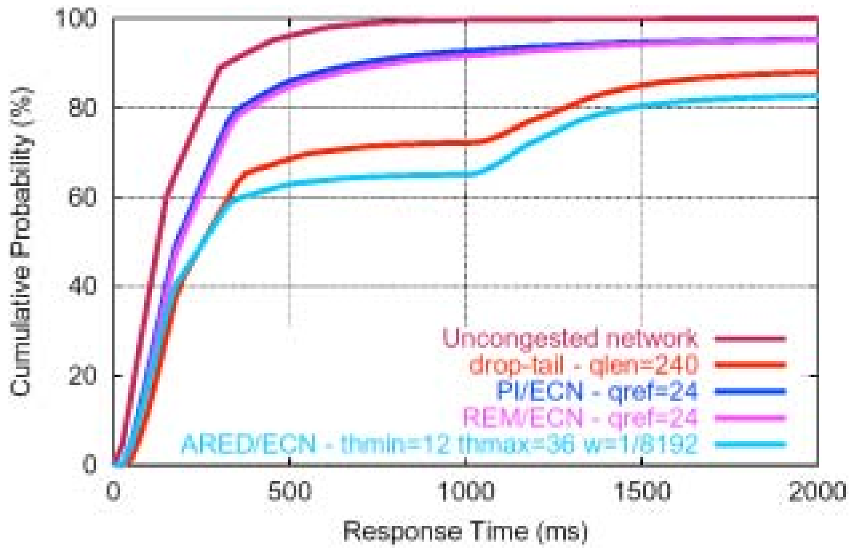


ECN Results — 90% Load
Comparison of all schemes



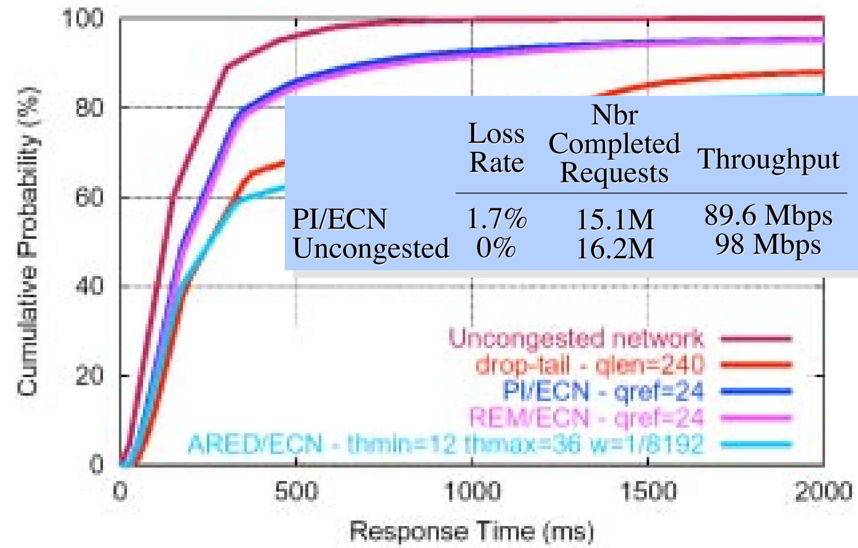
ECN Results — 98% Load

Comparison of all schemes



ECN Results — 98% Load

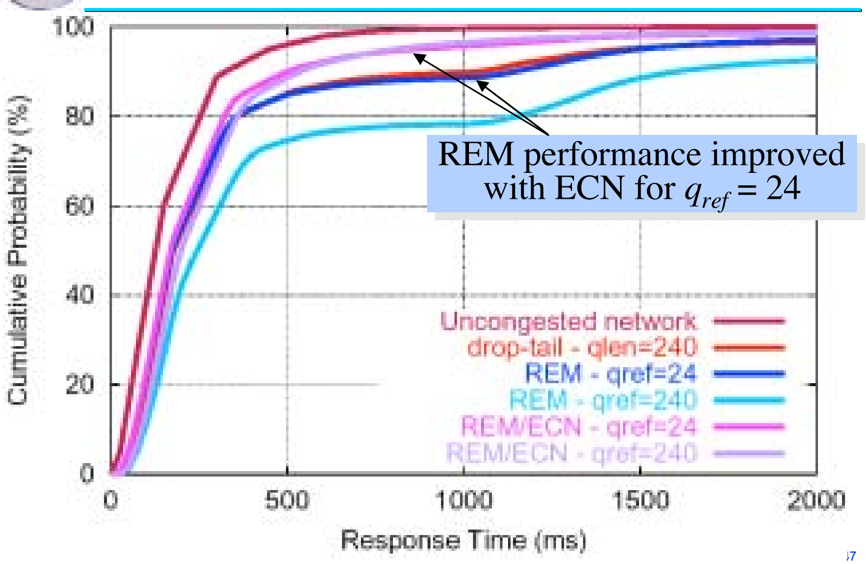
Comparison of all schemes



	Loss Rate	Nbr Completed Requests	Throughput
PI/ECN	1.7%	15.1M	89.6 Mbps
Uncongested	0%	16.2M	98 Mbps

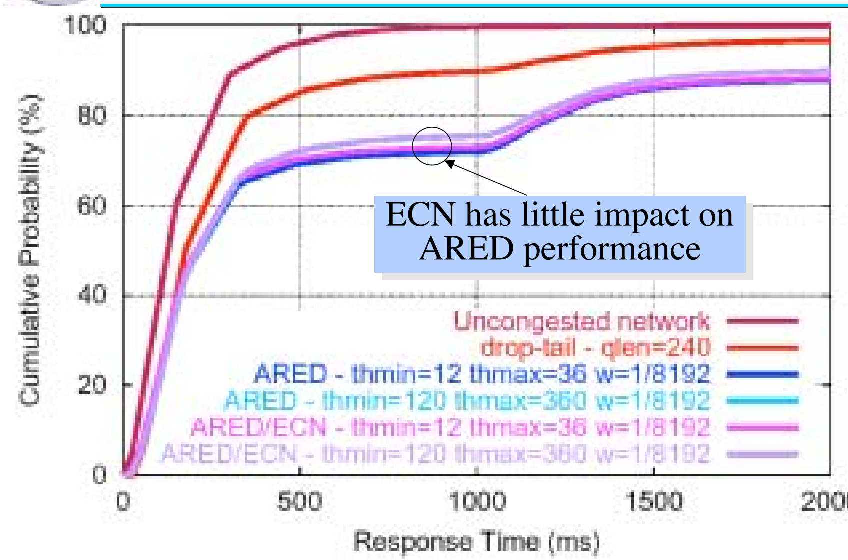
Impact of ECN on REM

Performance with/without ECN at 90% load

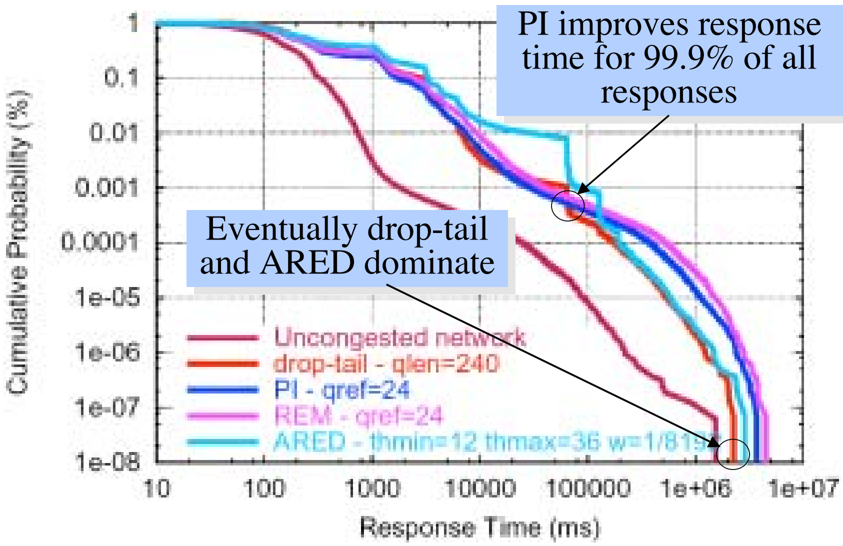


Impact of ECN on ARED

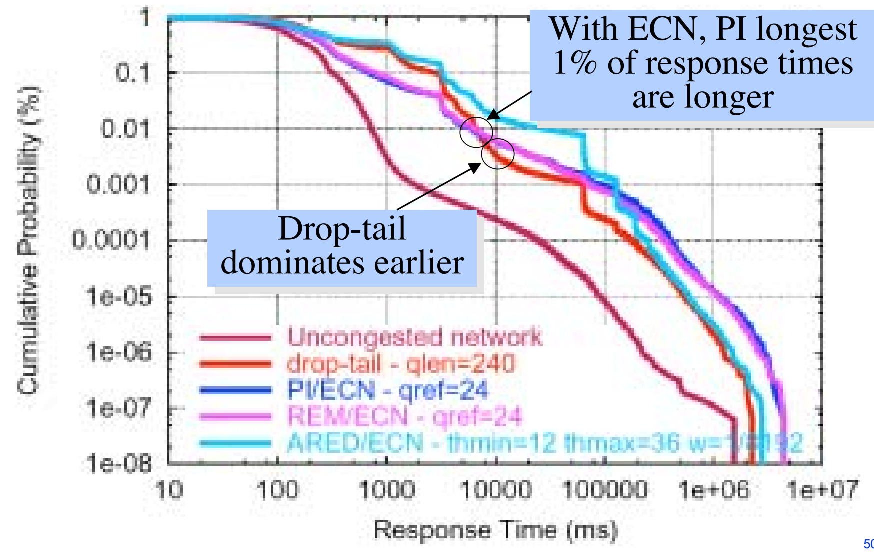
Performance with/without ECN at 90% load



Experimental Results — CCDFs
Comparison AQMs with drops at 98% load



Experimental Results — CCDFs
Comparison of AQMs with ECN at 98% load



Do AQM Schemes Work?
Summary

- For offered loads up to 80% of link capacity, no AQM scheme gives better performance than drop-tail FIFO
  - All give comparable response time performance, loss rates, and link utilization
- For offered loads of 90% or greater…
  - Without ECN, PI results in a modest performance improvement over drop-tail and other AQM schemes
  - With ECN, both PI and REM provide significant performance improvement over drop-tail
- ARED consistently results in the poorest performance
  - Often worse than drop-tail FIFO

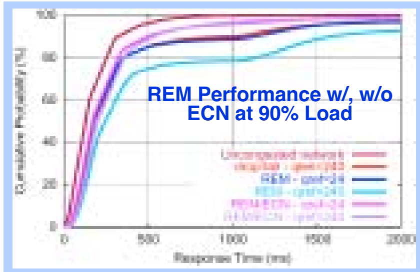
The Effects of AQM on the Web
Outline

- Review: Congestion control on the Internet today
- Router-based congestion control
  - Active Queue Management
  - Explicit Congestion Notification
- State of the art in active queue management (AQM)
  - Control theoretic *v.* traditional randomized dropping AQM
- Do AQM schemes work?
  - An empirical study of the effect of AQM on web performance
- Analysis of AQM performance
  - The good, the bad, and the ugly…

Discussion

Why does ECN improve REM more than PI?

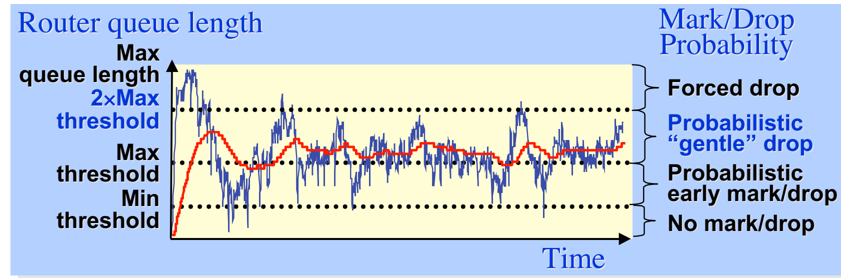
- Without ECN, REM drops more packets than PI
- REM causes more flows to experience multiple losses within a congestion window
  - Loss recovered through timeout rather than fast recovery



- In general ECN allows more flows to avoid timeouts
  - Thus ECN is ameliorating a design flaw in REM

Discussion

Why does ARED not benefit from ECN?

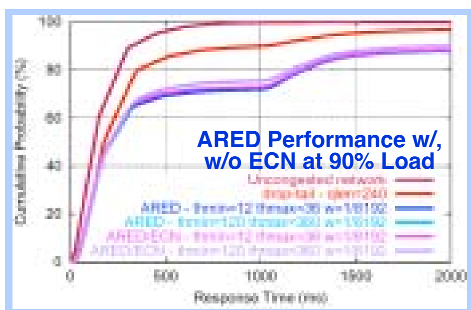


- ARED drops marked packets when average queue size is above max_{th}
- This is done to deal with potentially non-responsive flows
- We believe this policy is a premature optimization

Discussion

Why does ARED perform so poorly?

- PI and REM measure queue length in bytes
- By default RED measures in packets
  - But ARED does have a "byte mode"



- Drop/Mark probability in PI/REM biased by packet size
  - SYNs and pure ACKs have a lower drop probability in PI/REM
- Differentiating at the packet level is critical
  - Is it enough?

Discussion

Do AQM designs inherently require ECN?

- Claim: Differentiating between flows at the flow-level is important
- ECN is required for good AQM performance because it eliminates the need for short flows to retransmit (a significant fraction of their) data
  - With ECN, short flows (mostly) no longer retransmit data
  - But their performance is still hurt by AQM
- Why signal short flows at all?
  - They have no real transmission rate to adapt
  - Hence signaling these flows provides no benefit to the network and only hurts end-system performance

The Effects of AQM on the Web
Summary and Conclusions

- We emulated a peering point between two ISPs and applied AQM in ISP border routers
- We emulated the browsing behaviors of tens of thousands of users in a laboratory testbed
- No AQM scheme with or without ECN is better than drop-tail FIFO for offered loads up to 80% of link capacity
- For offered loads of 90% or greater there is benefit to control theoretic AQM but only when used with ECN

Future Work
Do AQM designs inherently require ECN?

- Claim: Differentiating between flows at the flow-level is important
- ECN is required for good AQM performance because it eliminates the need for short flows to retransmit (a significant fraction of their) data
  - With ECN, short flows (mostly) no longer retransmit data
  - But their performance is still hurt by AQM
- Why signal short flows at all?
  - They have no real transmission rate to adapt
  - Hence signaling these flows provides no benefit to the network and only hurts end-system performance
- How specific are these results to Web traffic?
  - How hard is it to experiment with "real" Internet traffic?

The UNIVERSITY *of* NORTH CAROLINA
at CHAPEL HILL

The Effects of Active Queue Management on Web Performance

The Good, the Bad, and the Ugly

Long Le, Jay Aikat, Kevin Jeffay, and Don Smith

November 2003

http://www.cs.unc.edu/Research/dirt