The UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

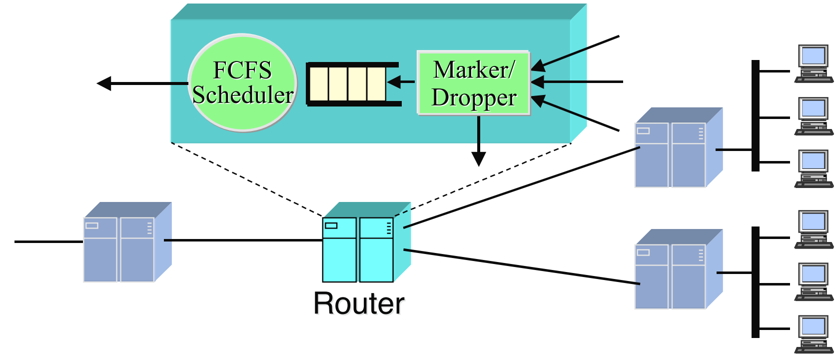
The Effects of Active Queue Management on Web Performance

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

SIGCOMM 2003

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

The Effects of Active Queue Management on Web Performance



- An AQM goal was minimizing delays for interactive applications such as web browsing [RFC 2309]
  - This is achieved by minimizing the average queue size in routers

The Effects of AQM on Web Performance
Overview

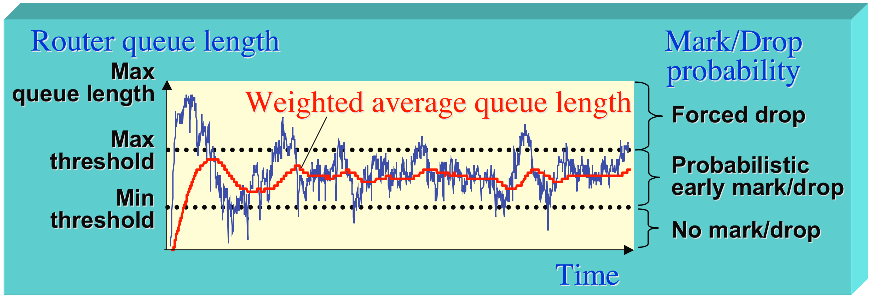
- We've conducted an empirical evaluation of the effects of three prominent AQM schemes…
  - PI, REM, and Adaptive/Gentle RED

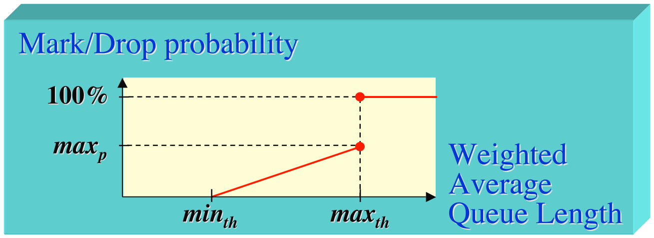
  …on the response time of web-like applications
  - AQM schemes evaluated with and without ECN
- For HTTP response times, we conclude:
  - No AQM scheme is better than drop-tail FIFO for offered loads up to 80% of link capacity
  - Above 90% of link capacity, PI and REM with ECN provide significant improvement over drop-tail
  - Adaptive/Gentle RED consistently results in the poorest performance (poorer than drop-tail)

The Effects of AQM on Web Performance
Outline

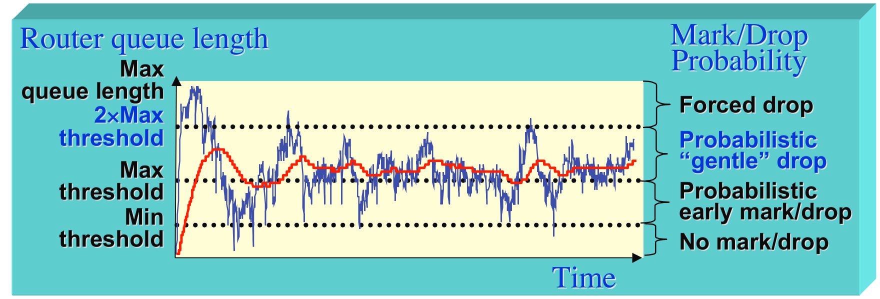
- Active queue management algorithms considered
  - ARED: Adaptive/Gentle Random Early Detection
  - PI: Proportional Integral controller
  - REM: Random Exponential Marking
- Experimental methodology
  - HTTP traffic model
  - Live simulation facility
  - Traffic generation method
- Experimental results
  - Results with packet drops
  - Results with ECN
- Conclusions

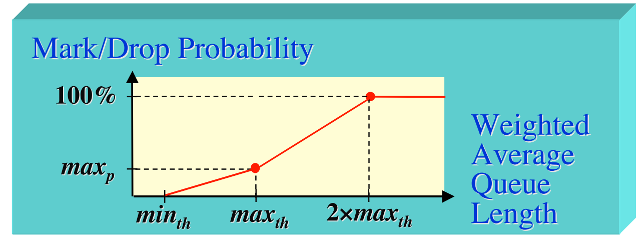
AQM Algorithms Considered
The original RED Algorithm



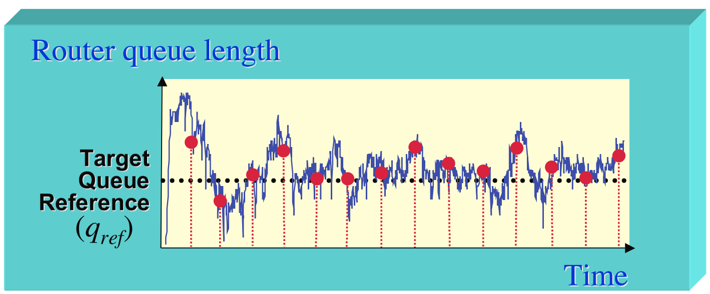


AQM Algorithms Considered
Adaptive/Gentle RED (ARED)



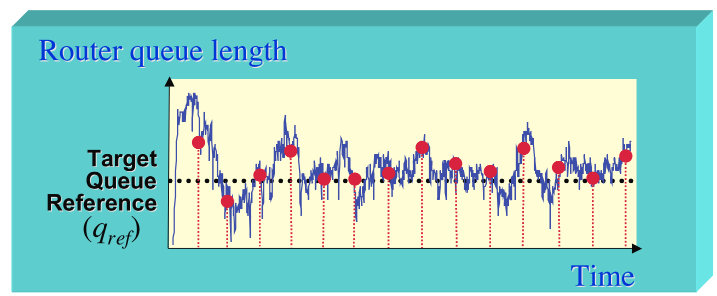


Control Theoretic AQM
The Proportional Integral (PI) controller



- 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

Control Theoretic AQM
Random Exponential Marking (REM)

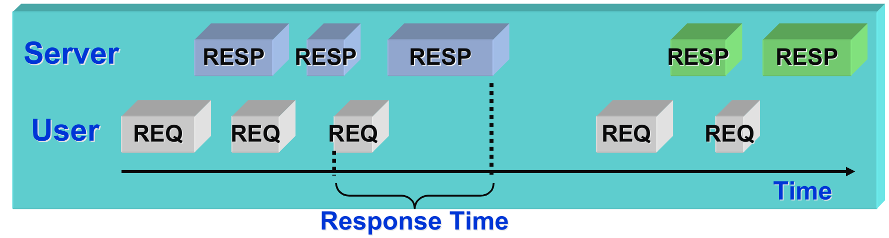


- REM is similar to PI (though differs in details)
- REM mark/drop probability depends on:
  - Difference between input and output rate
  - Difference between instantaneous queue length and target
  - $p(t) = p(t-1) + \gamma\,[\alpha\,(q(t) - q_{ref})) + x(t) - c]$
  - $prob(t) = 1 - \phi^{-p(t)}$, $\phi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

Experimental Methodology
Overview

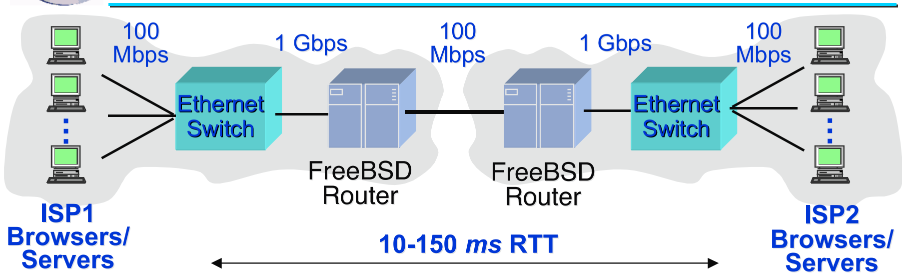
- Evaluate AQM schemes through "live simulation"
- Emulate the browsing behavior of a large population of users surfing the web in a laboratory testbed
  - Construct a physical network emulating a congested peering link between two ISPs
  - Generate synthetic HTTP requests and responses but transmit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

Experimental Methodology
HTTP traffic generation



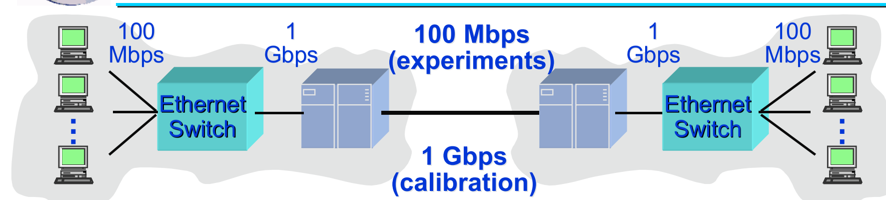
- Synthetic web traffic generated using the UNC HTTP model [SIGMETRICS 2001, MASCOTS 2003]
- Primary random variables:
  - Request sizes/Reply sizes
  - User think time
  - Persistent connection usage
  - Nbr of objects per persistent connection
  - Number of embedded images/page
  - Number of parallel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

Experimental Methodology
Testbed emulating an ISP peering link



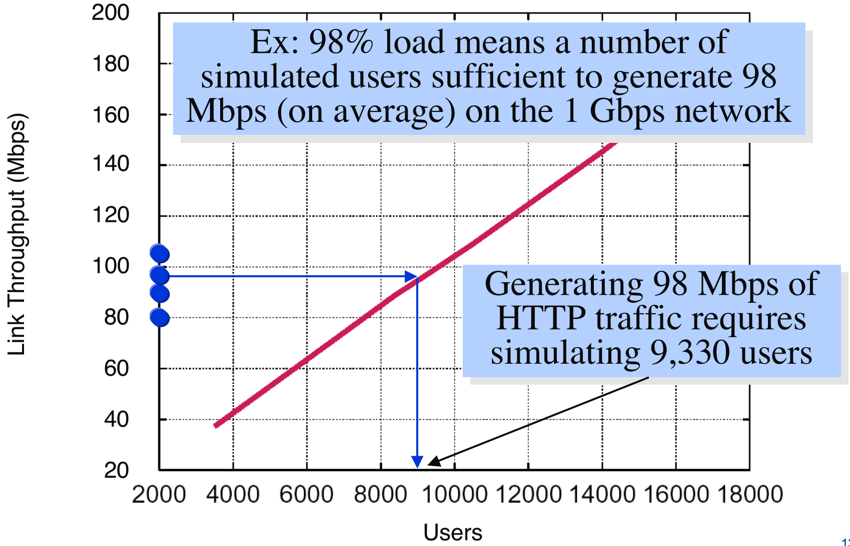
- AQM schemes implemented in FreeBSD routers using ALTQ kernel extensions
- End-systems either a traffic generation client or server
  - Use *dummynet* to provide *per-flow* propagation delays
  - Two-way traffic generated, equal load generated in each direction

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 linear in the number of simulated users — *i.e.*, that end-systems are not a bottleneck)

Experimental Methodology
1 Gbps network calibration experiments



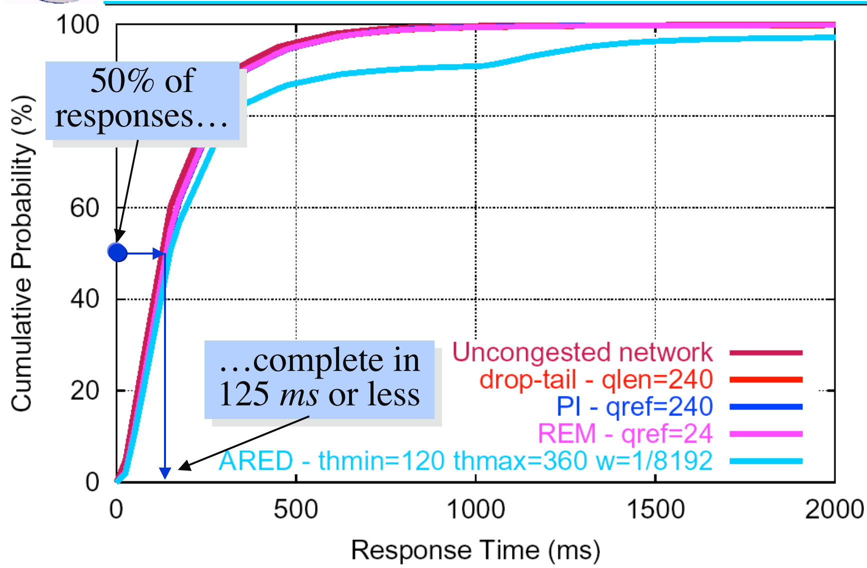
Experimental Methodology
Experimental plan

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

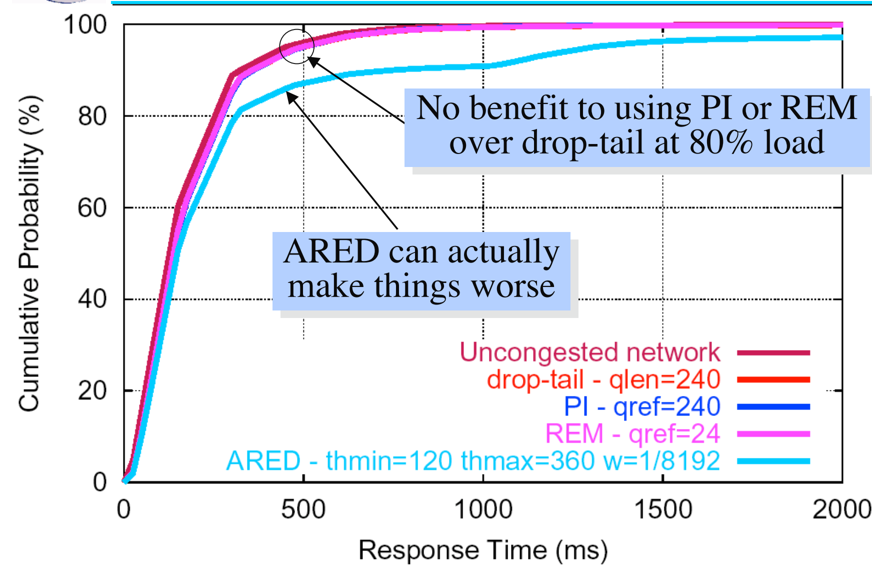
- Run experiments with ARED, PI, and REM using their recommended parameter settings at different offered loads
- Compare results with drop-tail FIFO at the same offered loads…
  - The "negative" baselines (the performance to beat)

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

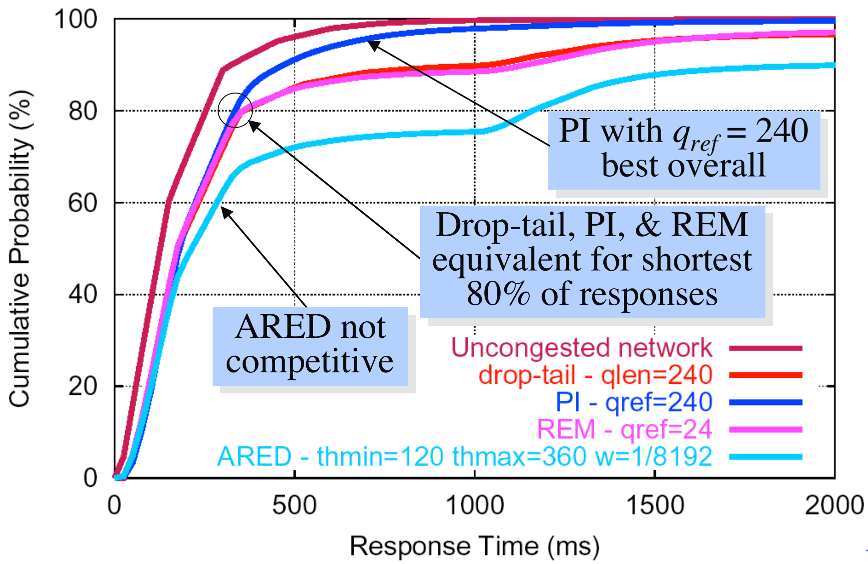
Experimental Results — 80% Load
Performance with packet drops



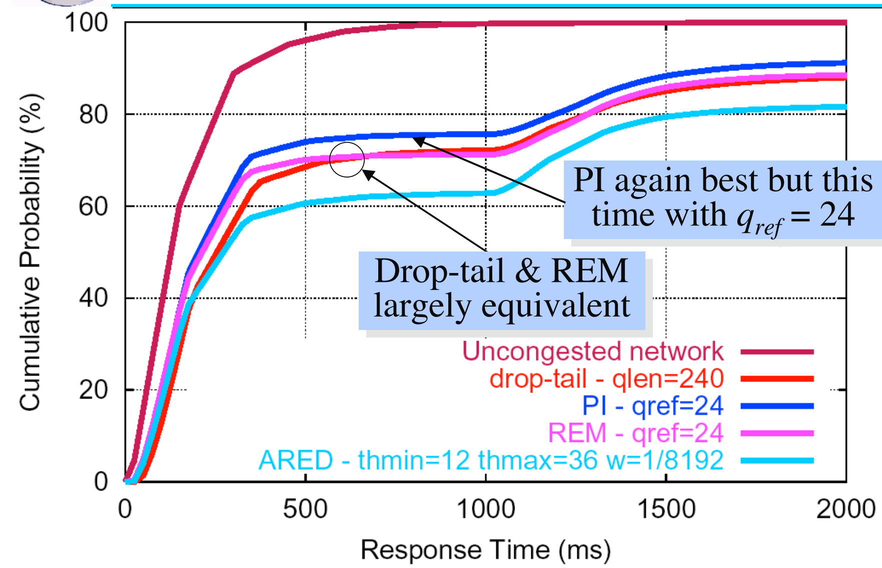
Experimental Results — 80% Load
Performance with packet drops



Experimental Results — 90% Load
Performance with packet drops



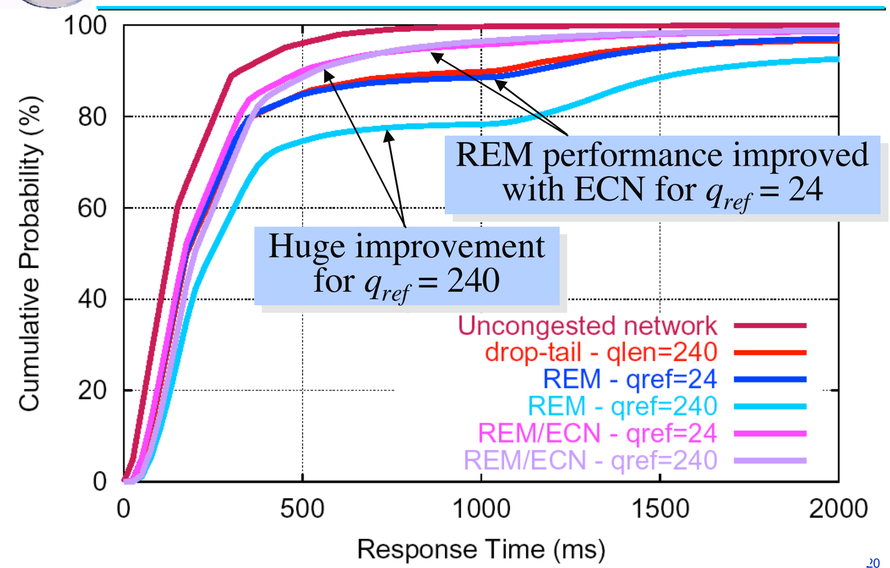
Experimental Results — 98% Load
Performance with packet drops



Performance With Packet Drops
Summary

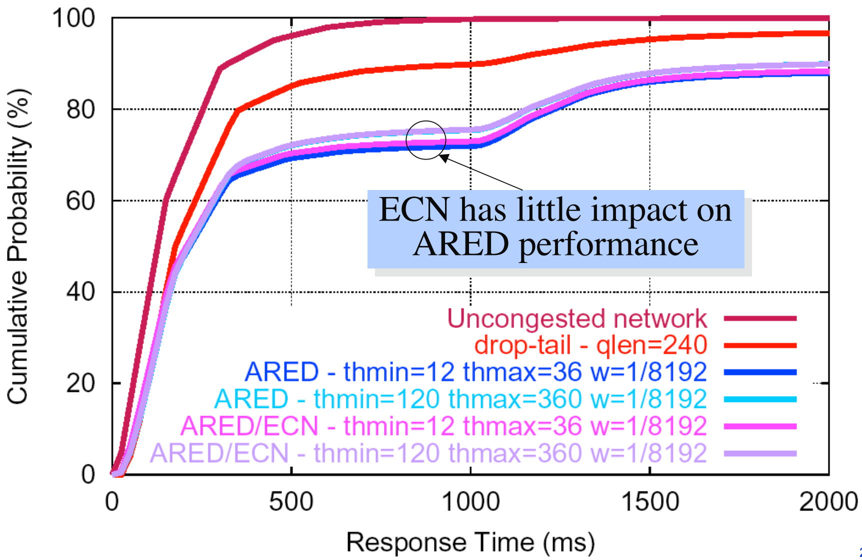
- At offered loads of 80% or below there is no benefit to employing PI or REM over drop-tail FIFO
  - All give comparable response time performance, loss rates, and link utilization
- There is a negative effect to employing ARED (at all loads)
  - Our attempts to tune ARED performance were unsuccessful
- At 90% and 98% loads PI outperforms drop-tail and REM
  - But best parameter settings are load sensitive

Experimental Results — REM
Performance with/without ECN at 90% load



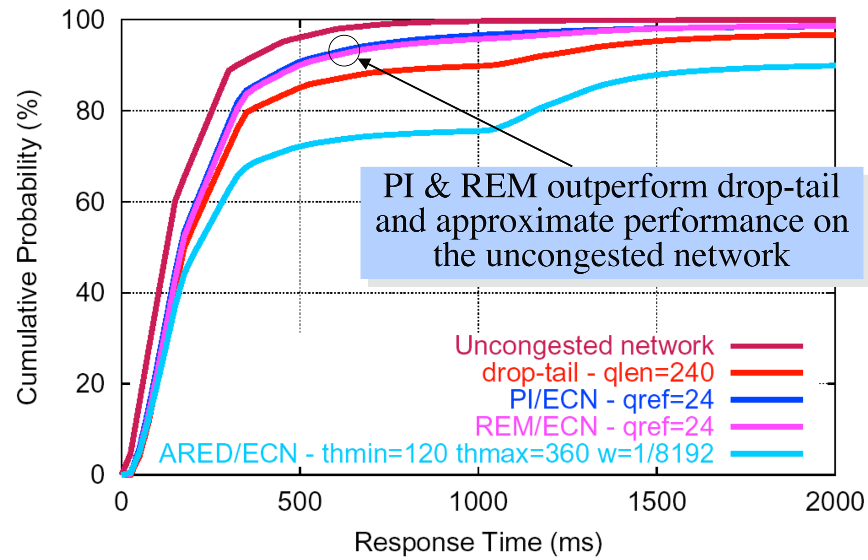
Experimental Results — ARED

Performance with/without ECN at 90% load



Experimental Results — 90% Load

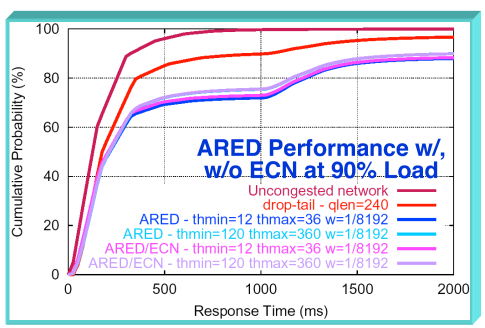
Performance with ECN



Discussion

1. Why does ARED perform so poorly?

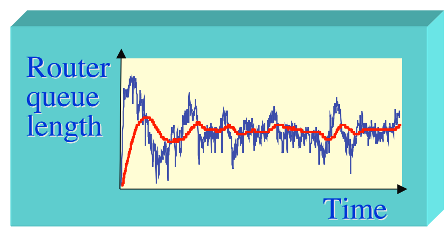
- PI and REM measure queue length in bytes
- By default RED measures in packets
  - But ARED does have a "byte mode"



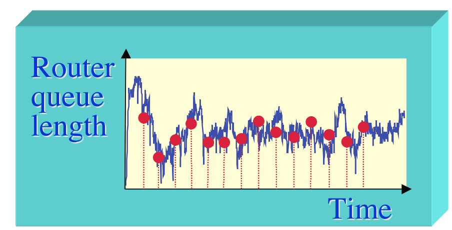
- Drop/Mark probability in PI/REM biased by packet size
  - SYNs and pure ACKs have a lower drop probability in PI/REM

Discussion

1. Why does ARED perform so poorly?



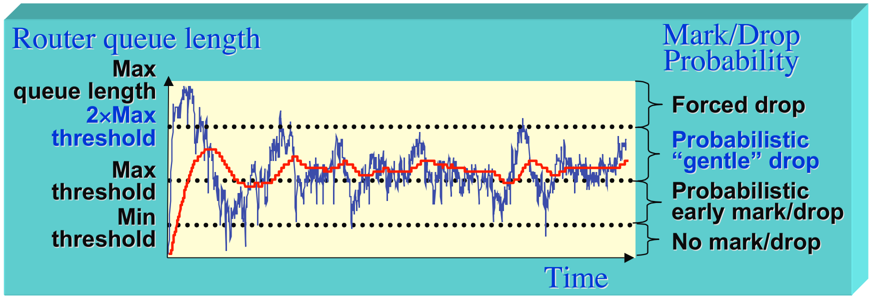
Weighted Queue Length (RED)



Instantaneous Queue Length (PI/REM)

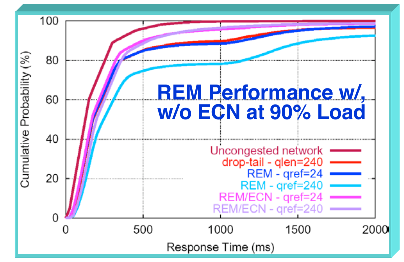
- ARED bases mark/drop probability on the (weighted) average queue length
- PI, REM use instantaneous measures of queue length
- ARED's reliance on the average queue length limits its ability to react effectively in the face of bursty traffic

Discussion
2. 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
3. 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
- Future work: Differential congestion notification
  - Don't signal short flows that can't adapt

The Effects of AQM on Web Performance
Conclusion

- No AQM is better than drop-tail FIFO for offered loads up to 80% of link capacity
- 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
  - (But is ECN improving PI & REM by ameliorating design limitations?)
- ARED consistently results in the poorest response time performance
  - Often worse than drop-tail FIFO

The UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

The Effects of Active Queue Management on Web Performance

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

SIGCOMM 2003

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