

Tuning RED for Web Traffic

*Mikkel Christiansen, Kevin Jeffay,
David Ott, Donelson Smith
UNC, Chapel Hill*

**SIGCOMM 2000, Stockholm
subsequently
IEEE/ACM Transactions on Networking
Vol. 9 , No. 3 (June 2001) pp 249 – 264.**

Presented by Bob Kinicki



Tuning RED Outline

- Introduction
- Background and Related Work
- Experimental Methodology
 - Web-like Traffic Generation
- Experiment Calibrations and Procedures
- FIFO and RED Results
- Conclusions

Introduction

- RFC2309 recommends **Active Queue Management [AQM]** for Internet congestion avoidance.
- **RED**, the best known AQM technique, has not been studied much for **Web traffic**, the dominant subset of TCP connections on the Internet in 2000.
- The authors use **response time**, a user-centric performance metric, to study short-lived TCP connections that model HTTP 1.0.

Introduction

- They model HTTP request-response pairs in a lab environment that simulates a large collection of *browsing users*.
- Artificial delays are added to a small lab testbed to approximate coast-to-coast US round trip times (RTT's).
- The paper focuses on studying **RED tuning parameters**.
- The basis of comparison is the effect of **RED** vs. **Drop Tail FIFO** on response time for HTTP 1.0.

Background and Related Work

- Authors review **RED** parameters (avg , $qlen$, min_{th} , max_{th} , w_q , max_p) and point to Sally Floyd guidelines.
- **RED** is effective in preventing congestion collapse when TCP windows configured to exceed network storage capacity.
- Claim by Villamizar and Song:: The bottleneck router queue size should be 1-2 times the bandwidth-delay product.
- **RED** issues (shortcomings) studied through alternatives: BLUE, Adaptive RED, BRED, FRED, SRED, and Cisco's WRED.
 - e.g. **FRED** shows that **RED** does not promote fair sharing of link bandwidth between TCP flows with long RTTs or small windows.

Background and Related Work

- ECN not considered in this paper.

The big deal - Most of the previous studies used small number of sources except **BLUE** paper with 1000-4000 Pareto on-off sources (but **BLUE** uses ECN).

- Previous tuning results include:
 - max_p is dependent on the number of flows.
 - router queue length stabilizes around max_{th} for a large number of flows.

Background and Related Work

- Previous analytic and simulation modeling at INRIA results:
 - **TCP goodput** does not improve significantly with **RED** and this effect is independent of the number of flows.
 - **RED** has lower mean queueing delay but much higher delay variance.
- Conclusion – research pieces missing include: Web-like traffic and worst-case studies where there are dynamically changing number of TCP flows with highly variable lifetimes.

Experimental Methodology

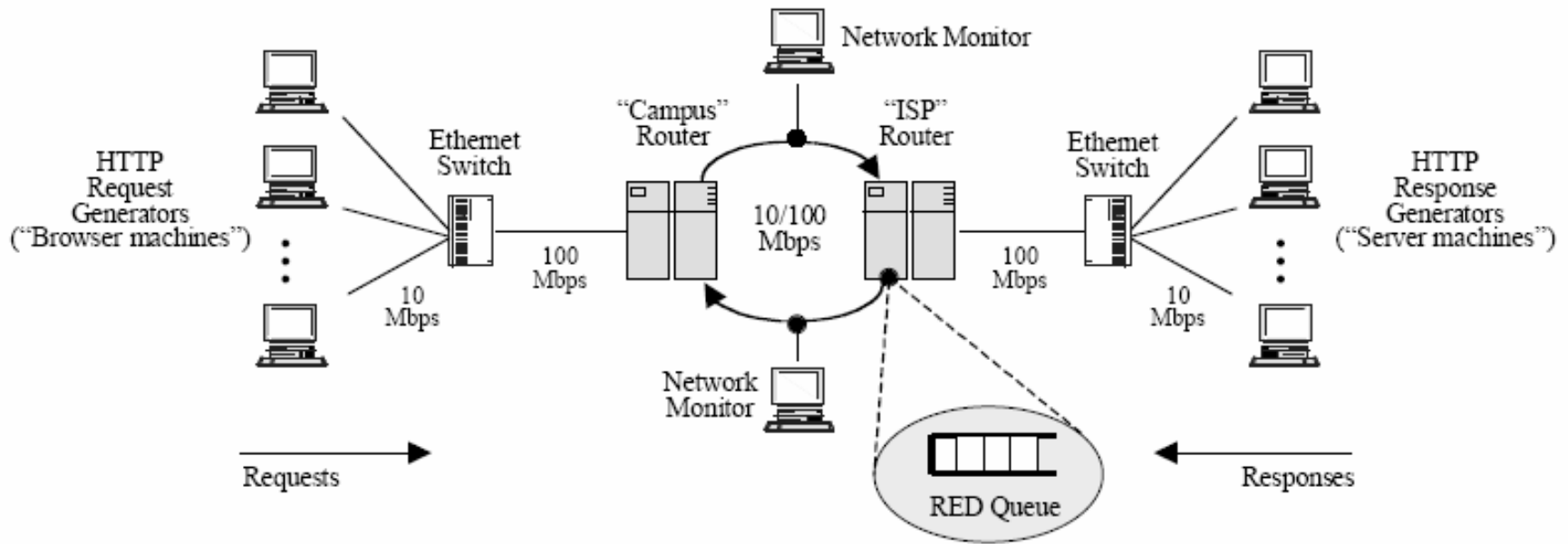


Figure 2: Experimental laboratory network diagram.

Experimental Methodology

These researchers used careful, meticulous, experimental techniques that are excellent.

- They use FreeBSD 2.2.8, ALTQ version 1.2 extensions, and *dumynet* to build a lab configuration that emulates full-duplex Web traffic through two routers separating Web request generators {*browser machines*} from Web servers.
- They emulate RTT's uniformly selected from 7-137 ms. range derived from measured data.
- FreeBSD default TCP window size of 16KB was used.
- A modified version of *tcpdump* is used to collect TCP/IP headers.

Web-like Traffic Generation

- The synthetic HTTP traffic for the experiments is based on Mah's Web browsing model [1995 data] that include:
 - HTTP request length in bytes
 - HTTP reply length in bytes
 - The number of embedded (file) references per page
 - The time between retrieval of two successive pages (user think time)
 - The number of consecutive pages requested from a server.

Web-like Traffic Generation

- The empirical distributions for all these elements were used in synthetic-traffic generators they built.
- The client-side request-generation program emulates behavioral elements of Web browsing.
- Important parameters include the size of server requests, the number of browser users (several hundred!!) the program represents and the user think time.
- A **new** TCP connection is made for each request/response pair (HTTP 1.0).
- Another parameter: number of concurrent TCP connections per browser user.

Experiment Calibrations and Procedures

1. They needed to insure that congested link between routers was the **primary bottleneck** on the end-to-end path.
2. They needed to guarantee that the offered load on the testbed network could be predictably controlled using the **number of emulated browser users** as a parameter to the traffic generator.

Experimental Methodology

- Monitoring tools:
 - At router interface collect: router queue size mean and variance, max queue size, min queue size sampled every 3 ms.
 - The machine connected to hubs forming links to routers uses a modified version of *tcpdump* to produce log of link throughput.
 - end-to-end measurements done on end-systems (e.g., response times)

Experimental Calibrations and Procedures

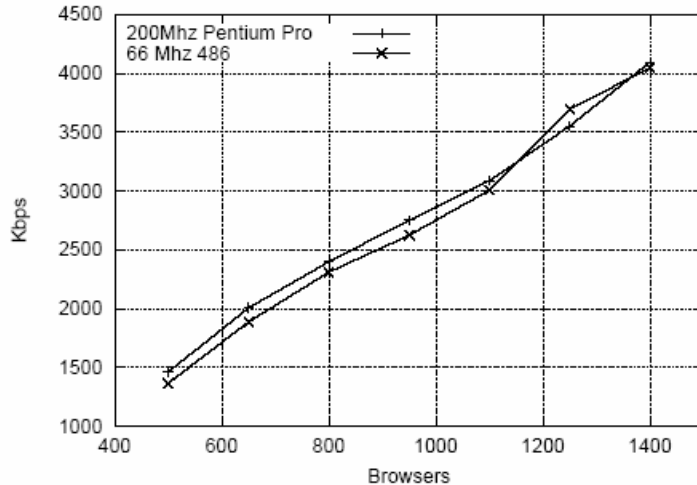


Figure 3: Offered load as a function of the number of simulated users on one machine.

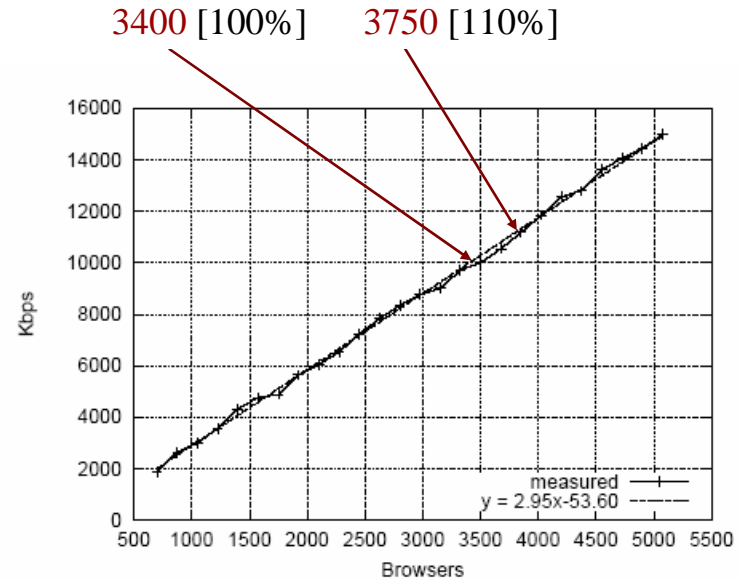


Figure 4: Offered load as a function of the number of simulated users on 7 machines.

Figure 3 and 4 show desired linear increases that imply no fundamental resource limitations. Note – these runs use a 100 Mbps link.

The authors were concerned about exceeding 64 socket descriptors limitation on one FreeBSD process. This limit was never encountered due to long user think times.

Experimental Calibrations and Procedures

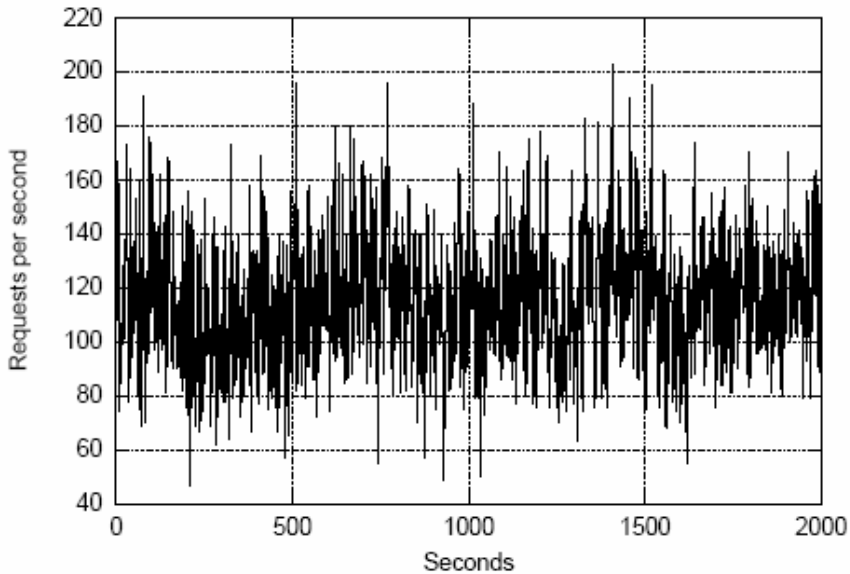


Figure 5: Requests per second from 3,500 users.

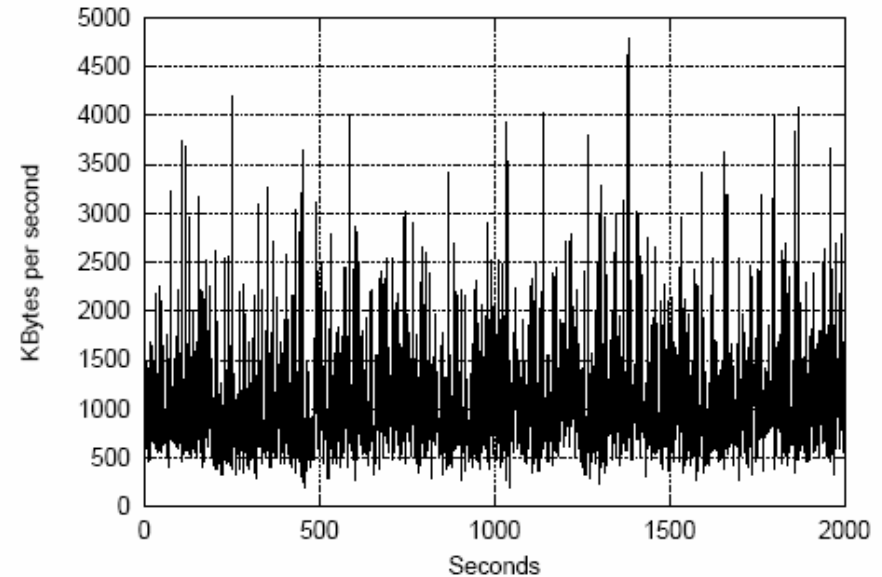


Figure 6: Bytes requested per second from 3,500 users.

Figures 5 and 6 show the highly *bursty nature* of requests by 3500 users during one second intervals.

Experimental Procedures

- After initializing and configuring the test-bed, the server-side processes were started followed by the browser processes.
- Each browser emulated an equal number of users chosen to place load on network that represent 50, 70, 80, 90, 98 or 110 percent of 10 Mbps capacity.
- Each experiments ran for 90 minutes with the first 20 minutes discarded to eliminate startup and stabilization effects.

Experiment Calibrations and Procedures

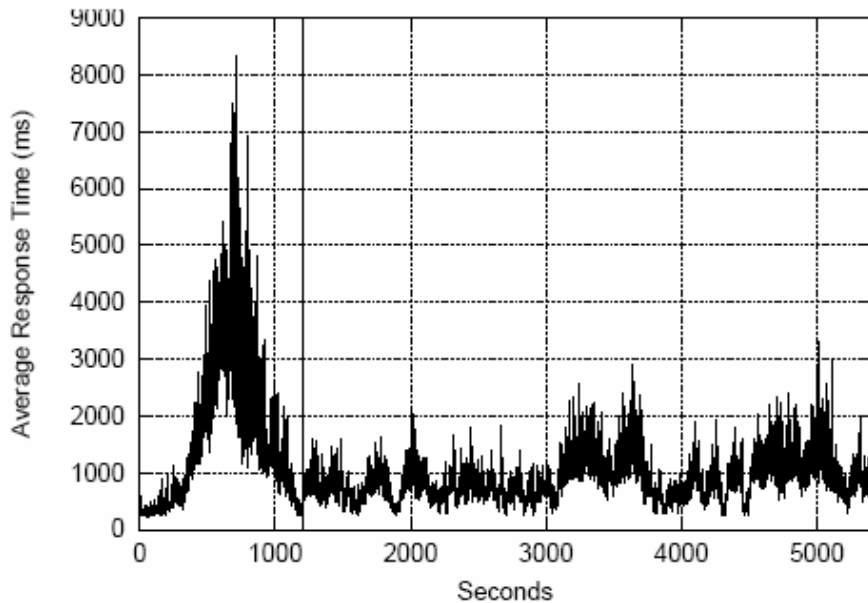


Figure 7: Average response time per second during an experiment. The plot includes the initial 20 minutes, where the traffic generators are started and stabilize.

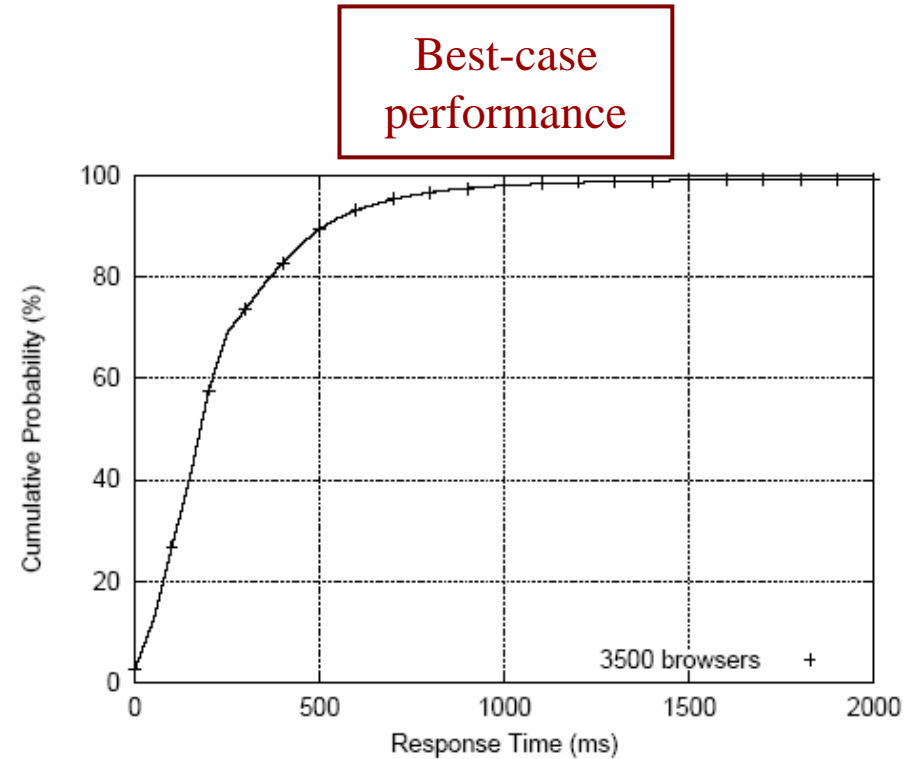


Figure 8: Cumulative response time distribution for 3,500 users on the unconstrained (100 Mbps) network.

Experimental Procedures

- Figure 8 represents the *best-case performance* for 3500 browsers generating request/response pairs in an *unconstrained* network.
- Since responses from the servers are much larger than requests to server, *only* effects on IP output queue carrying traffic from servers to browsers is reported.
- They measure: end-to-end response times, percent of IP packets dropped at the bottlenecked link, mean queue size and throughput achieved on the link.

FIFO Results [Drop Tail]

- FIFO tests run to establish a baseline.
- * For the critical FIFO parameter, *queue size*, the consensus is roughly 2-4 times the *bandwidth-delay product* (bdp)
 - mean min RTT = 79 ms.
 - + 10 Mbps congested link => 96 K bytes (bdp)
 - measured IP datagrams approx. 1 K bytes → 190 - 380 elements in FIFO queue to be within guidelines.

FIFO Results

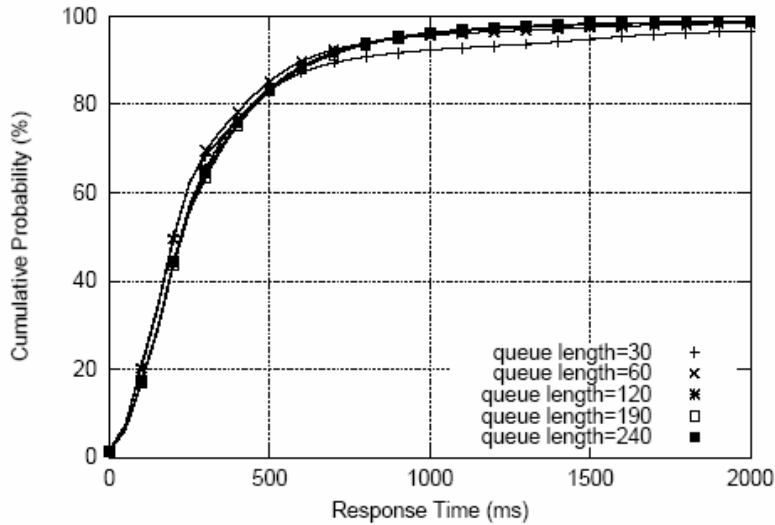


Figure 9a: FIFO performance at 80% load.

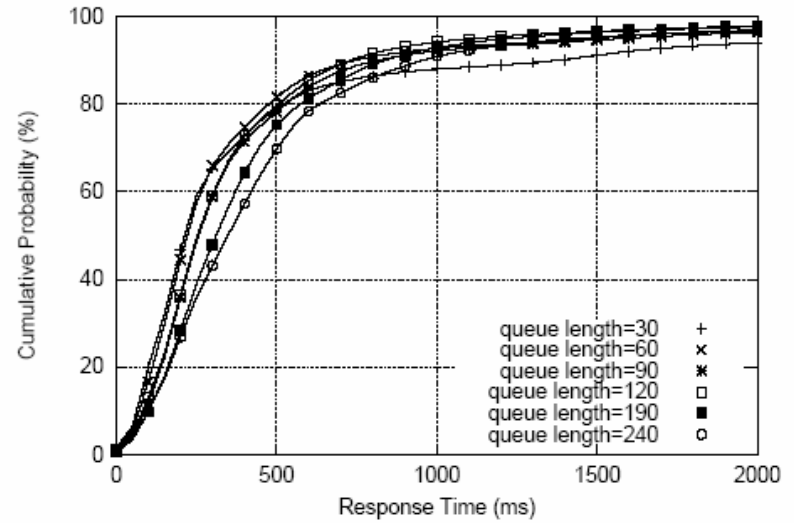


Figure 9b: FIFO performance at 90% load.

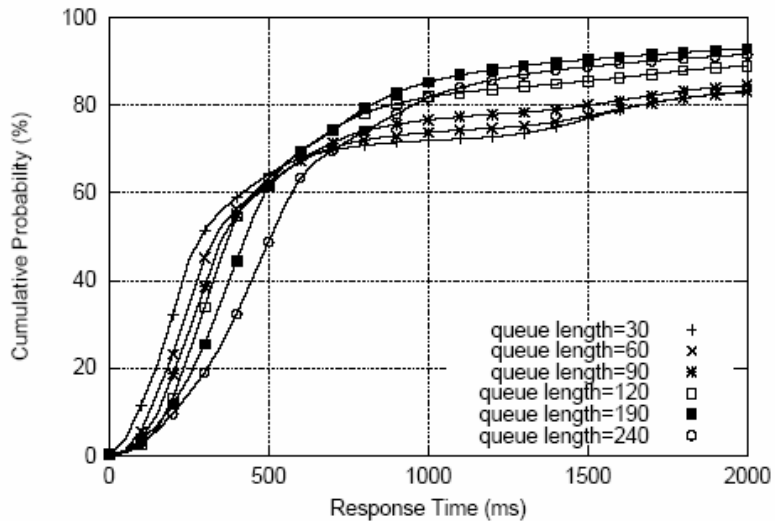


Figure 9c: FIFO performance at 98% load.

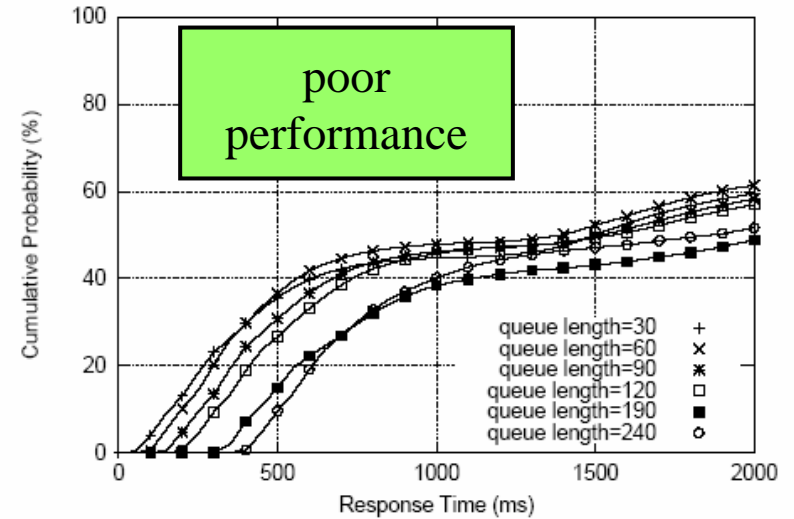


Figure 9d: FIFO performance at 110% load.

Appendix B

Table 1: FIFO results.

Load %	Queue Length	KB per sec.	% drops	Mean queue	Median resp. (ms)	% ≤ 1 sec.	1 < % ≤ 2 sec.	2 < % ≤ 3 sec.	% > 3 sec.
80	30	992	1.1	6.5	246	92.2	4.5	2.1	1.3
80	60	980	0.3	11.7	248	95.5	2.9	0.9	0.7
80	120	990	0.1	22.7	264	95.8	3.0	0.6	0.6
80	190	992	0.0	27.7	273	95.5	3.3	0.6	0.6
80	240	981	0.0	25.8	265	95.9	3.1	0.5	0.5
90	30	1107	2.2	9.9	258	87.7	6.2	3.7	2.5
90	60	1130	0.9	20.0	266	92.4	4.2	2.0	1.3
90	120	1164	0.3	40.3	298	93.7	4.0	1.3	1.0
90	190	1106	0.2	66.6	361	92.1	5.4	1.3	1.2
90	240	1179	0.3	85.7	397	89.9	6.8	1.6	1.7
98	30	1163	6.7	16.6	329	71.8	11.1	8.5	8.6
98	60	1177	6.2	41.6	375	73.5	9.3	7.8	9.4
98	120	1169	3.1	84.8	421	81.2	7.5	5.2	6.1
98	190	1166	1.3	119.2	478	84.2	8.5	3.3	4.0
98	240	1167	1.4	154.3	555	80.0	11.3	3.7	5.0
110	30	1183	18.9	22.6	1538	44.6	14.5	11.1	29.8
110	60	1189	16.4	52.4	1440	47.7	13.3	11.7	27.4
110	120	1188	17.0	112.3	1600	45.1	11.3	12.9	30.6
110	190	1188	19.3	183.0	2156	37.3	10.8	14.7	37.0
110	240	1188	16.5	231.7	1917	38.7	12.3	13.9	35.0

FIFO Results

- In Figure 9 a queue size of from 120 to 190 is a reasonable choice especially when one considers the tradeoffs for response time without significant loss in link utilization or high drops.
- At 98% (Figure 9c), one can see the tradeoff of using a queue length of 120. Namely, longer response times for shorter objects, but shorter response times for longer objects.

FIFO Results

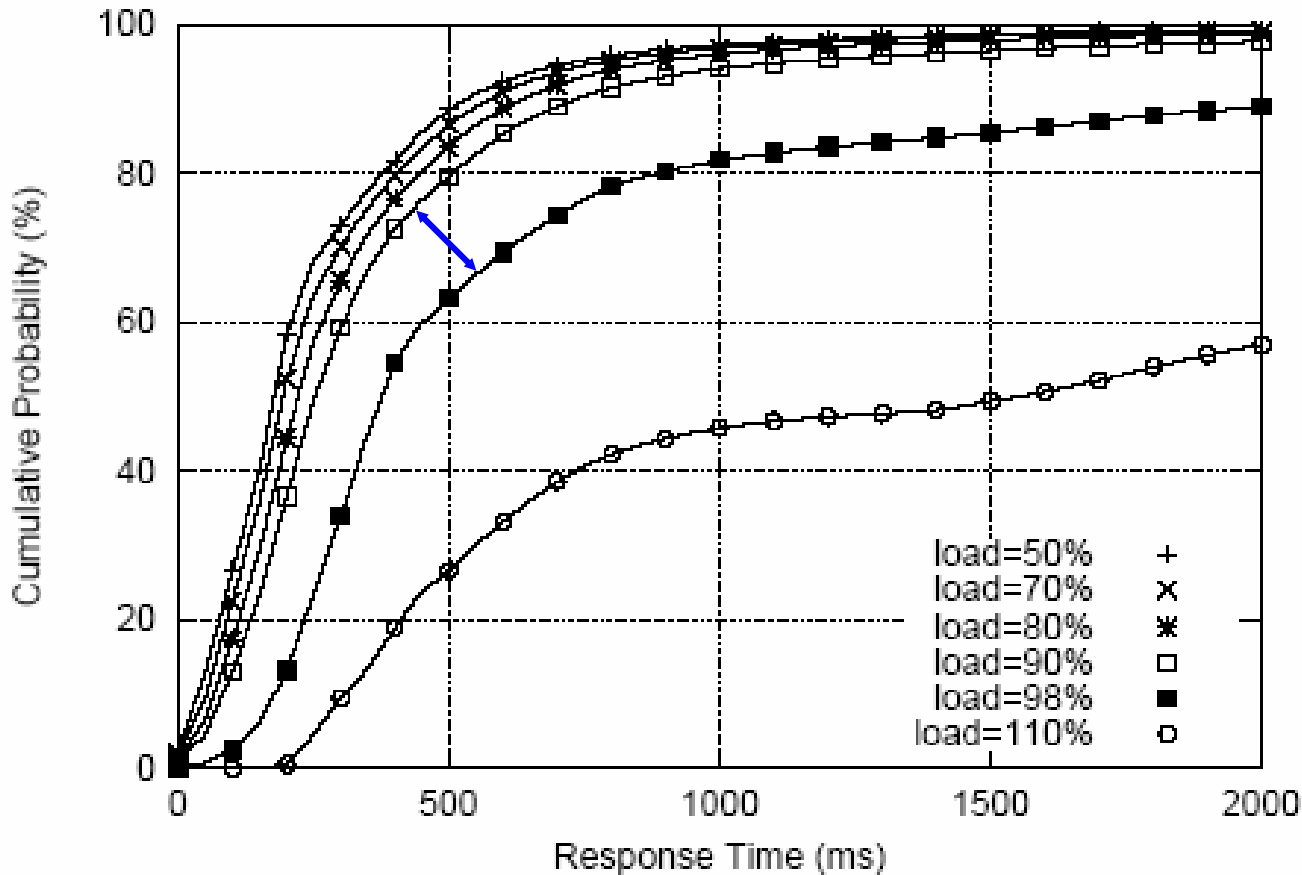


Figure 10: FIFO performance for different loads with a queue length of 120 elements.

Figure 10 FIFO Results

- At loads below 80% capacity, there is **no significant change** in response time as a function of load.
- Response time **degrades sharply** when offered load exceeds link capacity.

RED Results

- Experimental goal: *Determine the RED parameter settings that provide good performance for Web traffic.*
- Another objective is to examine the tradeoffs in **RED** tuning parameter choices.
- The FIFO results show complex tradeoff between response times for short responses and response times for longer responses.

RED Results

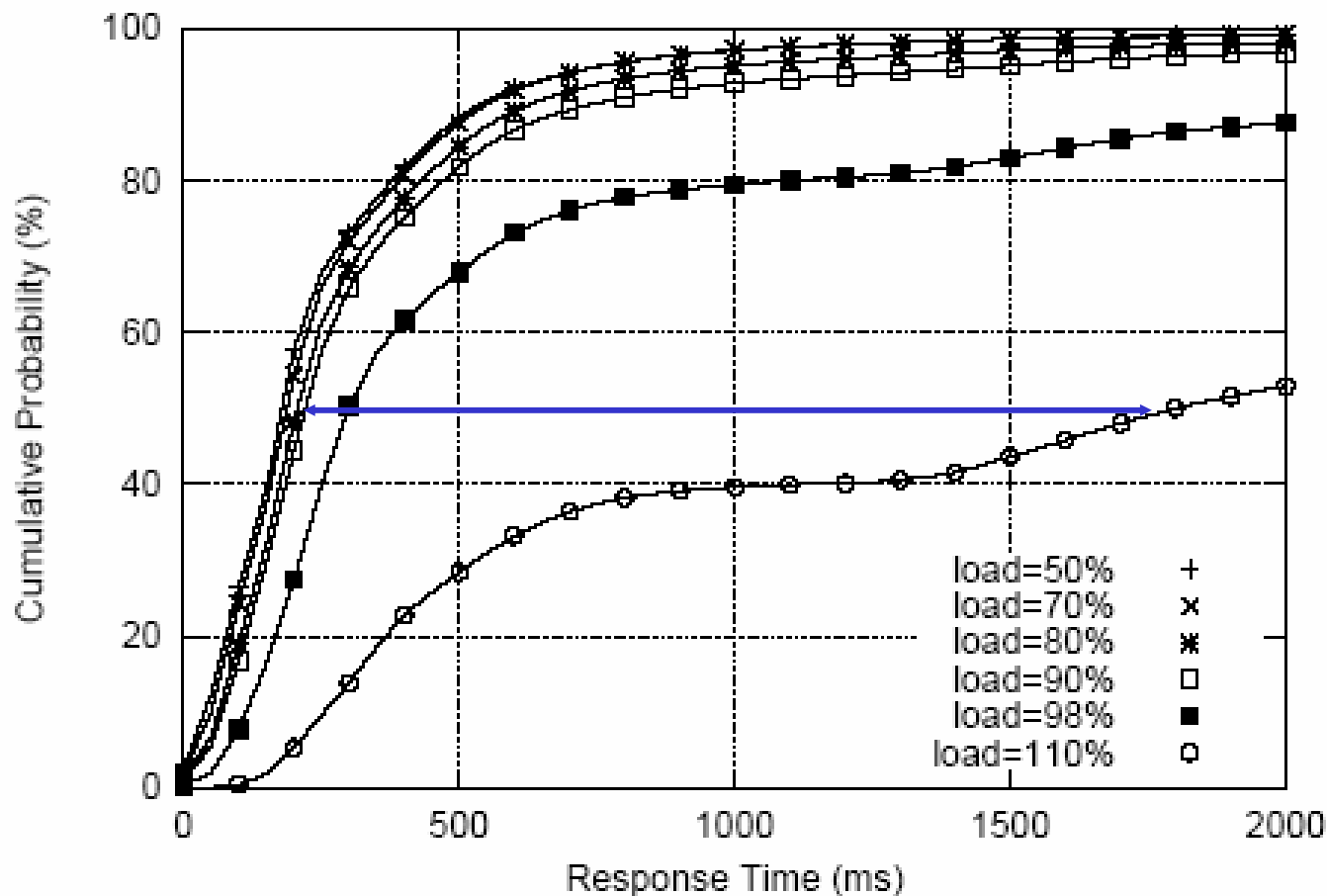


Figure 11: The performance of RED at different loads. $w_q=1/512$, $max_p=1/10$, $min_{th}=30$, $max_{th}=90$, $qlen=480$.

Figure 11 RED Results

The queue size was set to 480 to eliminate physical queue length ($qlen$) as a factor.

The figure shows the effect of varying loads on response time distributions.

- (min_{th}, max_{th}) set to (30, 90)
- The interesting range for varying RED parameters for optimization is between **90-110%** load levels where performance decreases significantly.

RED Results

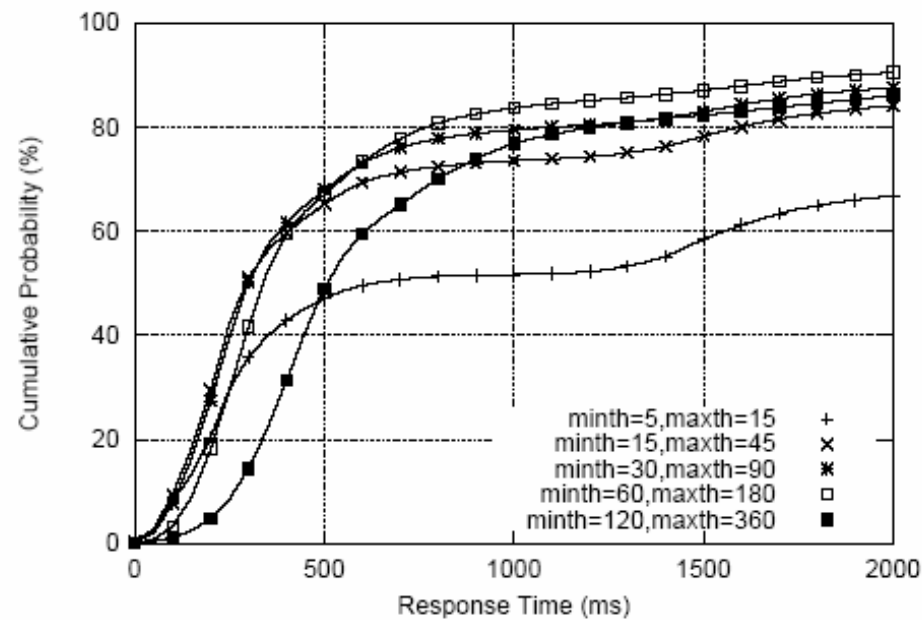
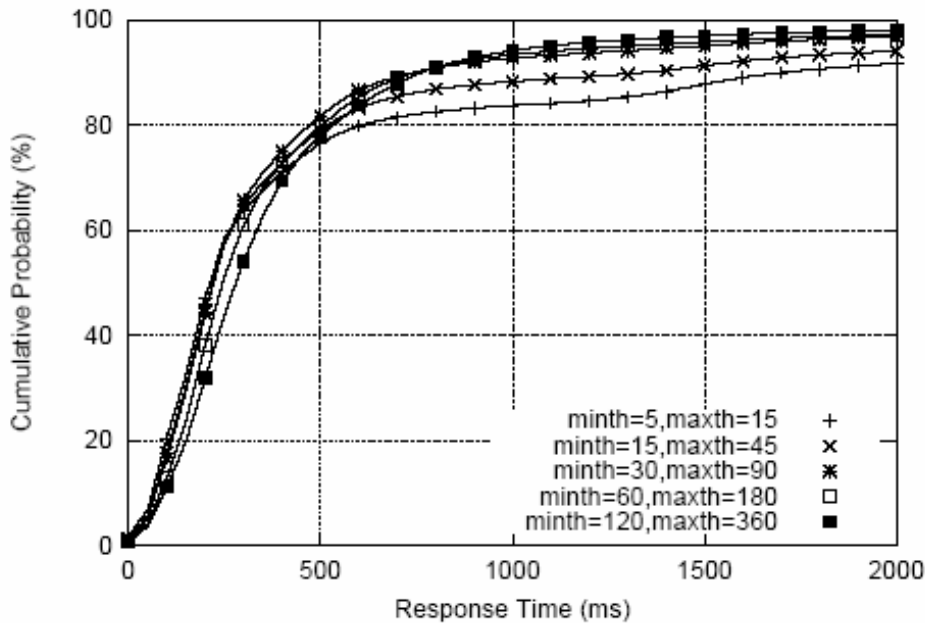
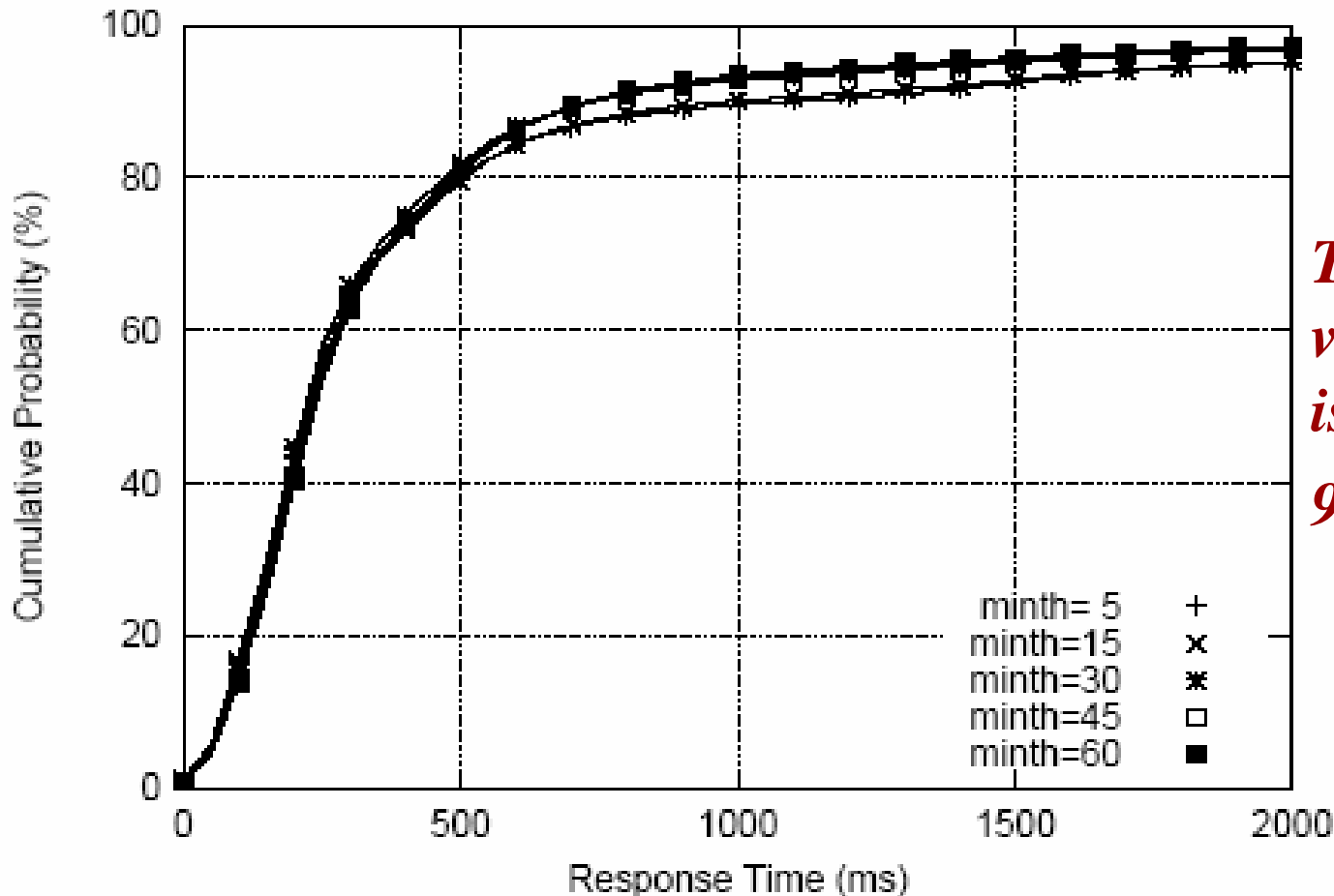


Figure 12a: Response time CDF for offered load at 90% of link capacity ($w_q=1/512$, $max_p=1/10$, $qlen=480$). **Figure 12b:** Response time CDF for offered load at 98% of link capacity ($w_q=1/512$, $max_p=1/10$, $qlen=480$).

Figure 12 RED Results

- The goal is to study min_{th} , max_{th} choices
 - The Floyd recommended choice (5, 15) yields bad performance at 90% load and poor performance at 98% load.
- (30, 90) or (60, 180) are the best choices!
- The authors prefer **(30, 90)** at 98% load.

RED Results



The effect of varying min_{th} is small at 90% load.

Figure 13: The effect of changing min_{th} . Load = 90% and $max_{th} = 90$, $w_q = 1/512$, $max_p = 1/10$, $qlen = 480$.

RED Results

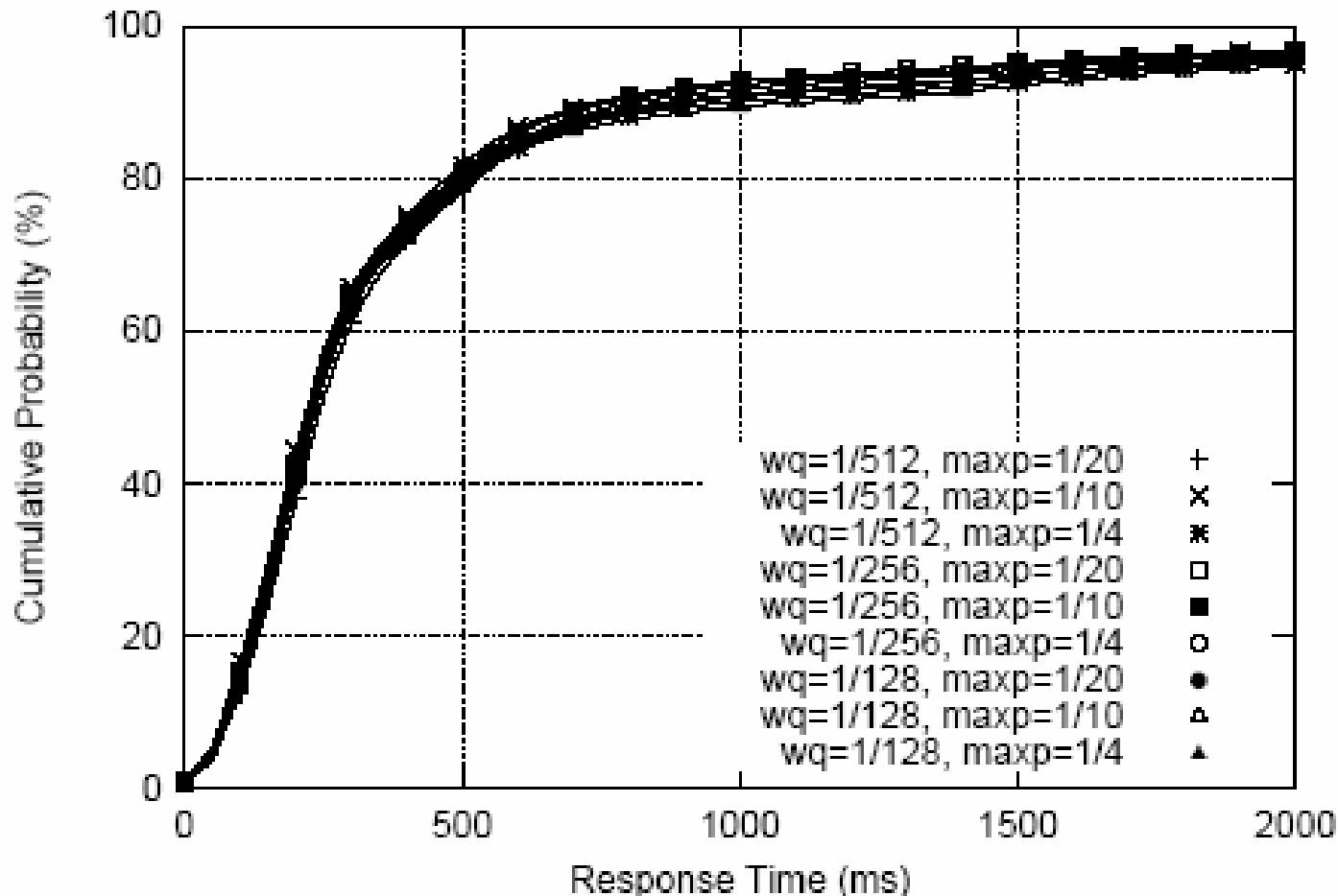


Figure 14: Results for different values of w_q and max_p .
Load = 90%, and $qlen = 480$, $min_{th} = 30$, $max_{th} = 90$.

Figure 14 RED Results

- $\max_p = 0.25$ has **negative impact** on performance – too many packets are dropped. Generally, changes in w_q and \max_p mainly impact *longer flows (the back part of the CDF)*.
- No evidence to use values other than recommended $w_q = 1/512$ and $\max_p = 0.10$

RED Results

Table 3: RED performance with recommended parameters and queue lengths.

Load %	Queue Length	KB/s	% drop	Mean queue	Median resp.(ms)	% ≤ 1 sec	1 < % ≤ 2 sec	2 < % ≤ 3 sec	% > 3 sec
90	480	1079	0.8	20.2	266	92.5	4.3	2.0	1.3
90	160	1093	1.1	22.2	278	91.2	4.7	2.4	1.7
90	120	1066	0.7	18.8	266	93.0	4.1	1.7	1.2
98	480	1164	4.1	39.4	345	79.2	8.2	6.3	6.3
98	160	1175	5.9	46.3	397	72.4	9.7	8.2	9.7
98	120	1171	5.5	44.3	377	74.2	9.2	7.7	8.9
110	480	1187	19.7	76.0	1846	39.4	12.9	12.1	35.5
110	160	1188	19.5	76.6	1864	39.1	13.0	12.2	35.7
110	120	1188	18.9	77.0	1840	39.3	13.2	12.5	34.9

120 is a good choice for queue length.

RED Results

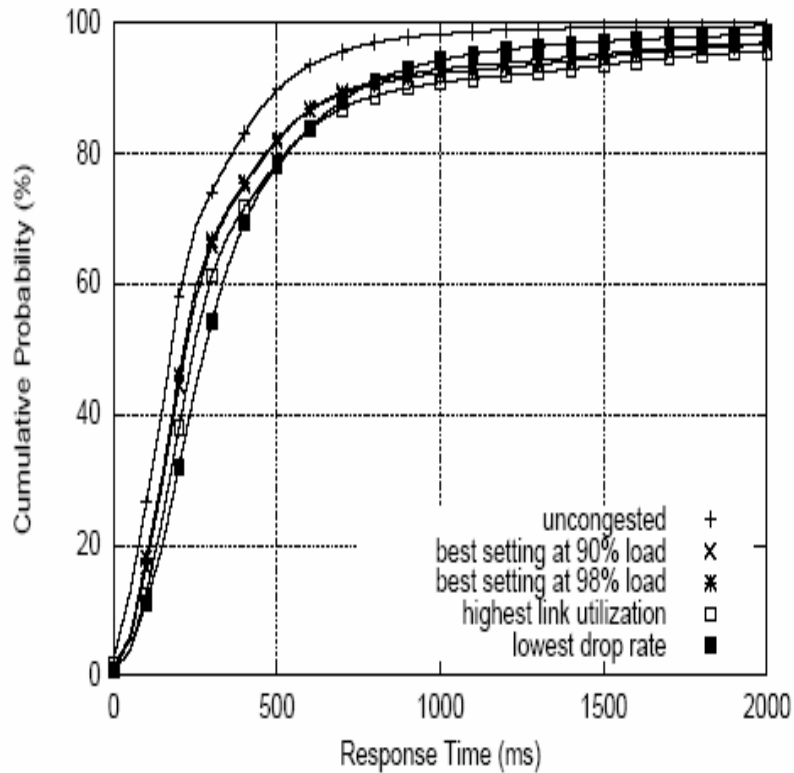


Figure 15a: “Good” RED parameter settings at 90% load.

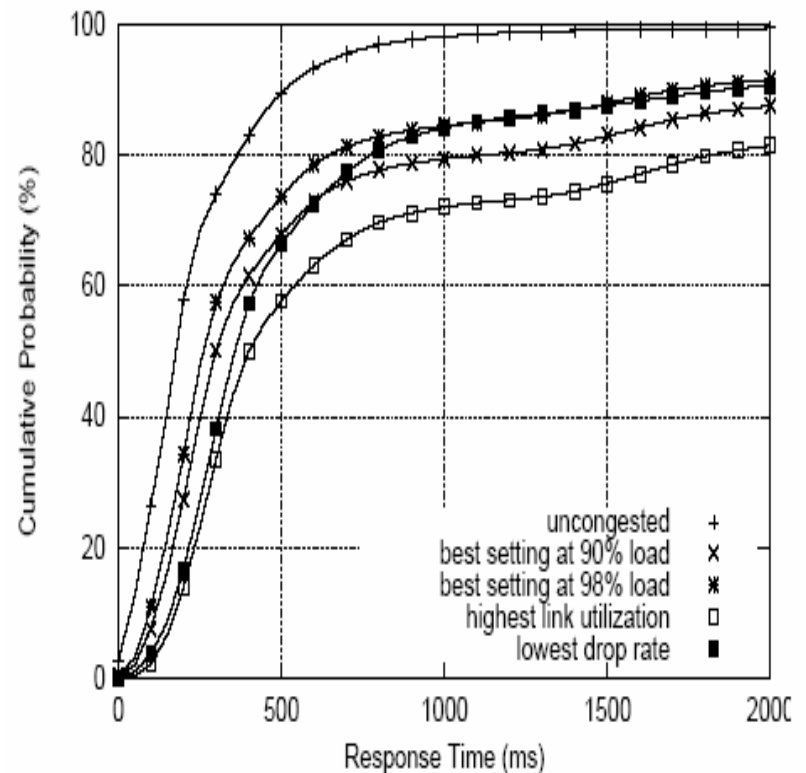


Figure 15b: “Good” RED parameters settings at 98% load.

RED Results

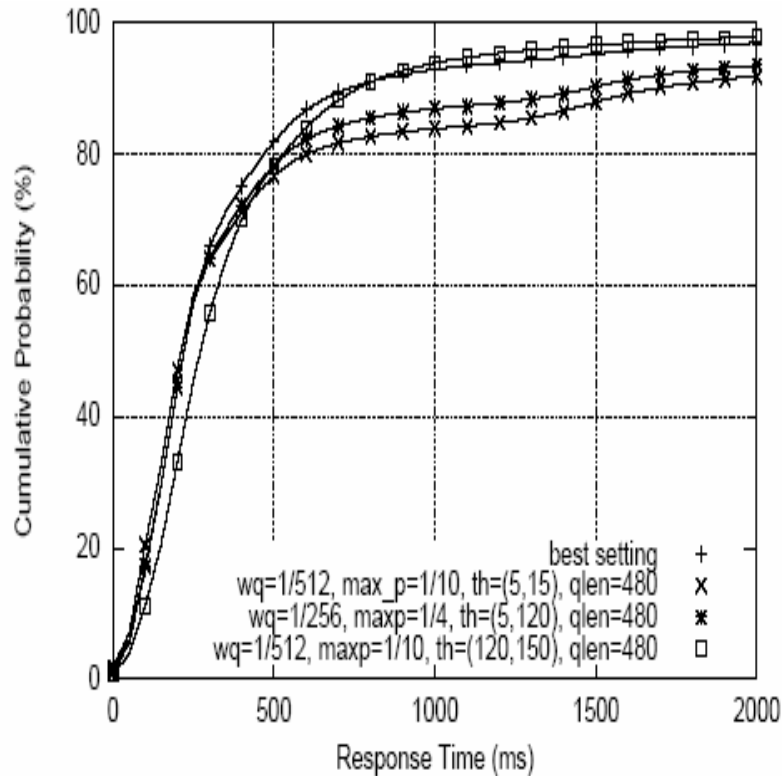


Figure 16a: "Bad" RED parameters settings at 90% load.

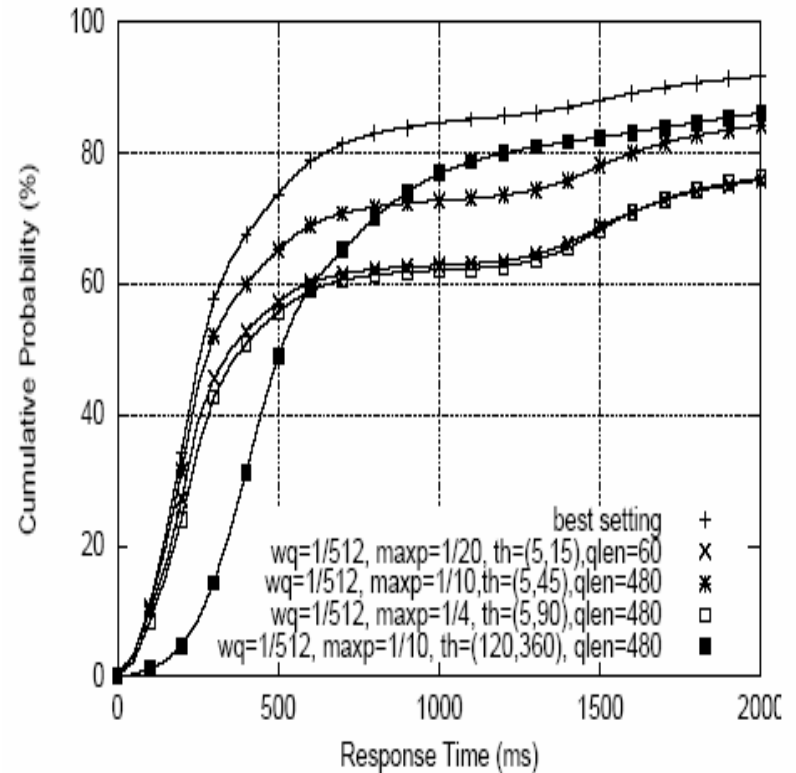


Figure 16b: "Bad" RED parameters settings at 98% load.

Figures 15 and 16 RED Results

- RED can be tuned to yield “best settings” for a given load percentage.
- At high loads, near saturation, there is a **significant downside potential** for choosing “bad” parameter settings

bottom line result- RED tuning is not easy!

RED Response Time Analysis

- This section added when paper went to journal.
- Detailed analysis of retransmission patterns for various TCP segments (e.g., SYN, FIN)
- This section reinforces the complexity of understanding the effects of RED for HTTP traffic.

RED Response Time Analysis

Table 5: Summary retransmission statistics for experiments with more detailed instrumentation.

Class of retransmission event	% of all TCP connections	
	$(min_{th}, max_{th}) = (5,15)$	$(60,180)$
No retransmissions	56.1	87.1
1 or more retransmissions	43.9	12.9
1 or more SYN segments	7.4	2.0
1 or more FIN segments	6.0	2.0
1 or more data segments	25.5	8.5
Combined SYN/FIN/data	5.0	0.4
Total TCP connections	439,979	460,022
Total segments lost	12.4%	2.4%

FIFO versus RED

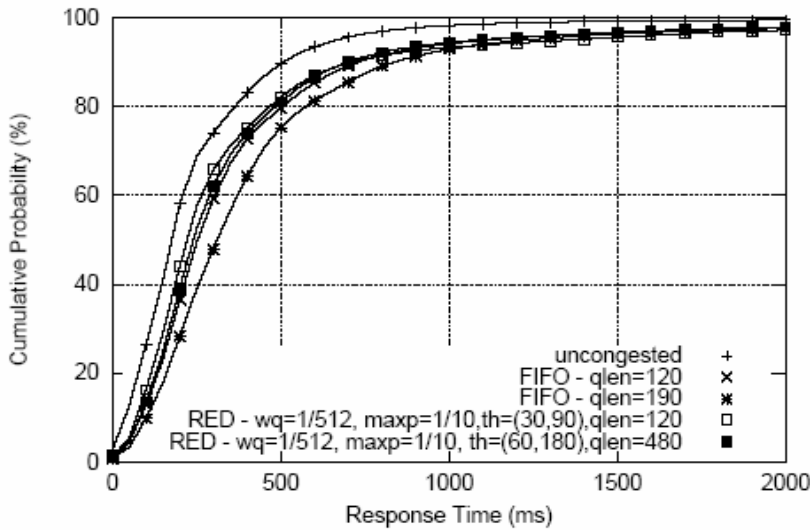


Figure 22a: FIFO and RED at 90% load.

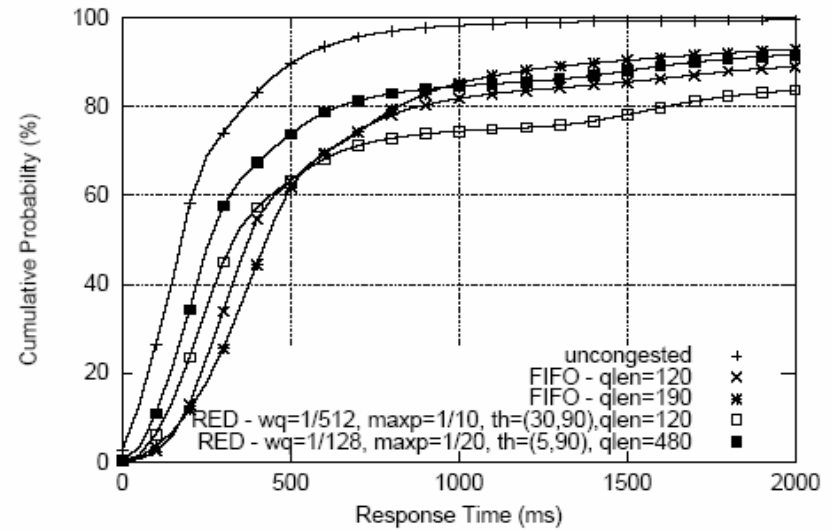


Figure 22b: FIFO and RED at 98% load.

*The only improvement for **RED** is at 98% load where careful tuning improves response times for shorter responses.*

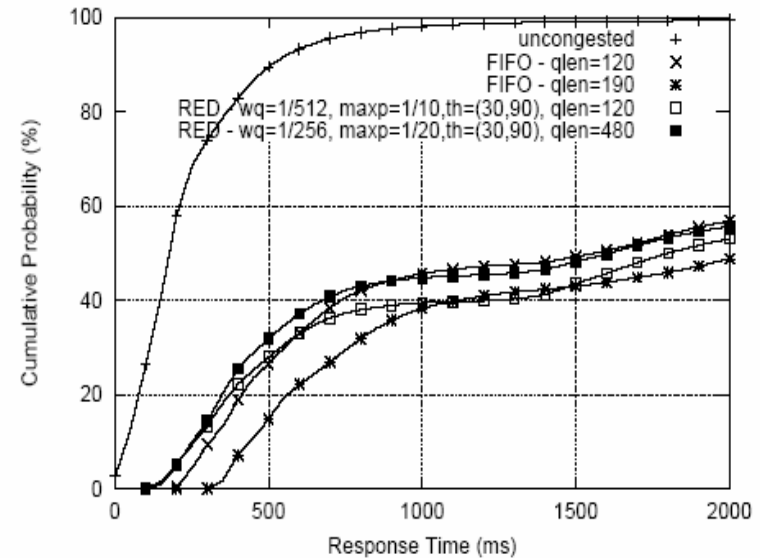


Figure 22c: FIFO and RED at 110% load.

Conclusions

- Contrary to expectations, there is little improvement in response times for **RED** for offered loads up to 90%.
- At loads approaching link saturation, **RED** can be **carefully** tuned to provide better response times.
- Above 90%, load response times are **more sensitive** to **RED** settings with a greater downside potential of choosing **bad parameter settings**.
- * There seems to be no advantage to deploying **RED** on links carrying only Web traffic.

Question: Why these results for these experiments?