Stability Criteria of RED with TCP Traffic

Thomas Ziegler[1] Christof Brandauer[2] Serge Fdida[3]
Thomas.Ziegler@ftw.at Christof.Brandauer@salzburgresearch.at Serge.Fdida@lip6.fr

[1] Telecommunications Research Center Vienna (ftw.), Maderstr. 4, 1040 Vienna, Austria
[2]Salzburg Research, J. Haringerstr. 5, 5020 Salzburg, Austria *
[3]Université Pierre et Marie Curie, Laboratoire Paris 6, Paris, France

Abstract

This paper systematically shows the caveats of non-optimal RED parameter setting and evaluates a quantitative model how to set RED parameters as a function of the scenario parameters bottleneck-bandwidth, round-trip-time and number of TCP flows. Using this model, it is shown by simulation that properly configuring RED for bulk-data and Web-like TCP traffic assuming knowledge of the scenario parameters is feasible and necessary. In order to show this necessity, the end-to-end performance decrease in case of non-optimal parameter setting is investigated.

Keywords: Congestion Control, TCP, Active Queue Management, RED

1 Introduction

The RED (Random Early Detection) queue management algorithm [1][2] for congestion avoidance in cooperation with end-to-end transport protocols has been widely discussed in the literature and is implemented in commercially available routers. RED uses the parameter-set {*minth, maxth, maxp, wq*} in order to probabilistically drop packets arriving at a router output port. If RED's average queue size (*avg*) is smaller than *minth* no packet is dropped. If *minth* < *avg* < *maxth,* RED's packet-drop probability varies between zero and *maxp*. If *avg* > *maxth*, each arriving packet is dropped. The average queue size is computed as the exponentially weighted moving average of the instantaneous queue size with weight parameter *wq*.

Recent simulations [3][4][5] with FTP-like traffic have shown that achieving RED's control-goal of making the average queue size converge between *minth* and *maxth* without heavy oscillations is sensitive to parameter setting. Oscillations are harmful as they cause periods of link underutilization when the instantaneous queue size equals zero followed by periods which can not be controlled by RED anymore when the average queue size exceeds *maxth* or the instantaneous queue approaches the total buffer size. In a DiffServ environment, oscillations may decrease RIO's [6] ability to preferentially drop out-of-profile packets against in-profile packets.

The problem of parameterizing RED means setting four inter-dependent RED parameters (*minth*, *maxth*, *maxp* and *wq*) properly as a function of three scenario parameters (bottleneck bandwidth, number of TCP flows, round-trip-time). If one of the RED parameters is not set properly convergence (i.e. bounded oscillation of the average queue size between *minth* and *maxth*) can not be achieved - see section 4. Note that configuring a RED gateway requires additional criteria to be taken into account, which are not considered in this paper as they are not of importance for stability. For instance, the setting of *wq* has also major influence on the RED gateway's ability to absorb traffic bursts; setting *minth* high enough is important for high link utilization in transient state.

* This work is partially funded by the IST project Aquila.

Section 2 summarizes related research; section 3 describes the simulation setup and section 4 illustrates the RED parameter dependencies. In section 5 a quantitative model for the setting of maxp, maxth and minth as developed in a companion paper [7] is summarized. Section 6 evaluates the model by simulations with Web-like TCP traffic and investigates the end-to-end performance of different RED parameter settings. Finally, section 7 concludes this paper.

2 Related Research

Early publications mentioning the topic of parameter setting give qualitative recommendations ignoring some dependencies of RED parameters on scenario parameters. As a rule of thumb, [1] [9] recommend to set *minth* to five packets and *maxth* to at least three times *minth*. The exact value of *minth* has to depend on the burstiness of the arrival process, link capacity, propagation delay and maximum buffer size. Additionally, it is remarked that the optimum setting of *minth* and *maxth* balances a trade-off between high queueing delay and poor link utilization. [1] mentions that the difference between *minth* and *maxth* should be greater than the typical increase of the average queue size in one round-trip-time. Without giving quantitative guidelines, the latter two statements imply that the difference between *minth* and *maxth* should be directly proportional to the bandwidth*delay product of the scenario. Based on the observation that steady state drop rates for TCP flows rarely exceed 5% in realistic scenarios, [9] recommends to set *maxp* to 0.1. Finally, it is recommended to set *wq* equal to 0.002 in order to balance a trade-off between setting *wq* too low and thereby responding too slowly to transient congestion and setting *wq* too high implying the incapability to filter out transient bursts. Based on measurement results and without developing a quantitative model, [10] recommends to set the buffersize with RED gateways to the bandwidth*RTT product in order to avoid link underutilization.

A first step to quantitatively model TCP with RED in order to provide recommendations for RED parameter settings has been made in [11]. This paper uses the TCP model proposed in [13] to set wq as a function of the scenario parameters. [12] proposes a quantitative model how to set *wq* based on RED's response to a unit-step input signal without modeling the feedback system as done in [11]. Independently of each other and using different approaches quantitative models for all RED parameters relevant for stability have been published in [7] and [14]. [15] compares the end-to-end performance of RED, gentle RED and Tail Drop. [16] shows that RED-like mechanisms tend to oscillate in the presence of two-way TCP traffic.

3 Simulation Settings

Abbreviations used throughout this paper:

- *B*: buffer size at bottleneck in packets
- *C*: bottleneck capacity link in Mbps
- L: bottleneck capacity in mean packets per second
- *D*: delay of Bottleneck link in ms
- *d*: total propagation delay in ms
- *N*: number of flows

Simulations presented in this paper haven been performed with ns-2 [17]. In all simulations *N* TCP flows start at random times between zero and 10 seconds of simulation time. Hosts 1-3 act as sources, hosts 4-6 act as sinks. Host 1 starts *N*/3 TCP flows to host4, host2 starts *N*/3 TCP flows to host5, host3 starts *N*/3 TCP flows to host6. TCP data-senders at host2 are of type TCP-SACK, all others are TCP-Reno. Packet sizes are uniformly distributed with a mean of 500 bytes. The simulated network is shown in figure 1. All 500 Mbps links have drop-tail queue management; buffer sizes are set sufficiently high to avoid packet loss. The link between router1 and router2 uses RED queue management. Packets are discarded solely at the bottleneck-link from router1 to router2.

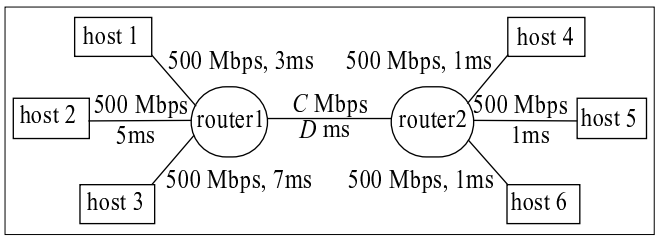

Figure 1 *simulated network*

RED is operated in packet mode as earlier simulations did not show any difference regarding the convergence behavior of the queue with RED in byte and packet mode. The "mean packet-size" parameter of RED is set to 500 bytes. In all queue size over time figures the *avg* is plotted with a fat line, the instantaneous queue size is plotted with a thin line. All simulations have been repeated several times in order to ensure convergence of results.

4 Dependency of RED Queue on Scenario Parameters

This section illustrates that simple rules of thumb, proposing constant RED parameter settings, are not sufficient to make the queue converge between minth and maxth in scenarios with infinite length bulk-data TCP flows. Contrary, the RED parameters relevant for stability (maxp, maxth-minth, wq) have to be set as a function of bottleneck bandwidth, RTT and number of TCP flows. As shown in this section, maxp determines RED's equilibrium point (i.e. the infinite time average of the RED queue size) which should at least in scenarios with bulk-data TCP traffic only stay close to (minth+maxth)/2. The difference between minth and maxth determines the amplitude of the oscillation of the queue size around the equilibrium point. Goal is to set maxth-minth high to avoid extensive oscillations and on the other hand as small as possible to keep queueing delay low. Finally, setting the wq parameter too small (i.e. making the average queue size track the instantaneous queue size too slowly) for a given scenario provides another cause for oscillation around the equilibrium point. In order to show the system behavior under the simpler steady state conditions, FTP like infinite bulk data TCP flows are simulated in this section. In section 6 realistic Web traffic will be used.

4.1 Constant maxp

Simulation1: varying bottleneck bandwidth

graph	C	D	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	0.5	100	100	8	33	50	10	0.002
b	5	100	100	42	167	250	10	0.002
c	20	100	100	100	400	600	10	0.002
d	50	100	100	100	400	600	10	0.002

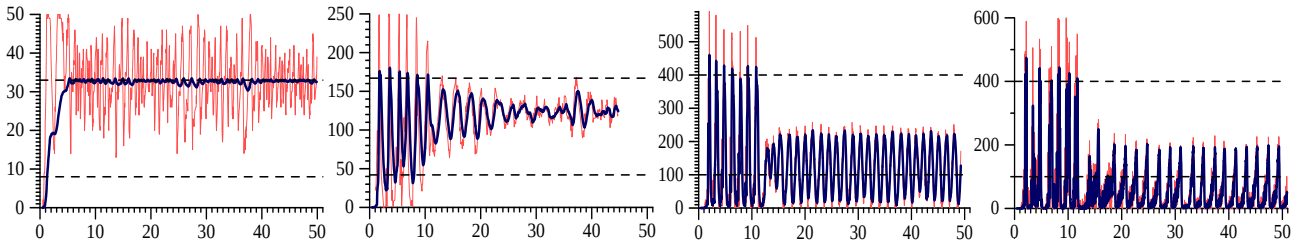

Figure 2 *a-d, inst. and average queue size over time*

The aggressiveness of an aggregate of TCP flows depends on the per flow bandwidth*RTT product as defined by C*RTT/N. The smaller the per flow bandwidth*RTT product, the higher the aggressiveness, and the higher maxp has to be set in order to make the equilibrium point stay close to (minth+maxth)/2. Simulation 1 varies the bottleneck capacity. As the increasing bandwidth causes the per-flow bandwidth*delay product to increase, the aggressiveness of the TCP flows and thereby the average queue size decreases from subfigure a to d. In figure a *maxp* is too small, causing convergence of *avg* to *maxth* and thus forced packet drops (i.e. packet drops not controlled by RED's drop function ranging from 0 to maxp). In figures c and d *maxp* is too high, causing *avg* to oscillate around *minth* resulting in poor link utilization as the queue is often empty.

Simulation2: varying *RTT*

graph	C	D	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	5	1	100	8	33	50	10	0.002
b	5	50	100	21	83	125	10	0.002
c	5	100	100	42	147	250	10	0.002
d	5	200	100	83	333	500	10	0.002

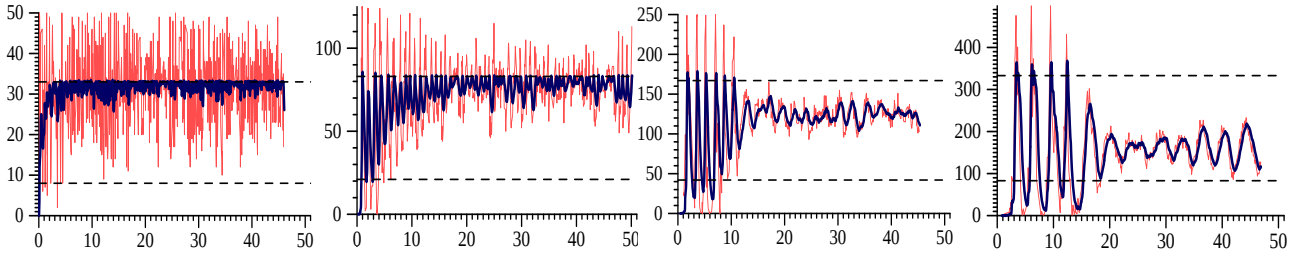

Figure 3 *a-d, inst. and average queue size over time*

Another parameter for the aggressiveness of a TCP aggregate is the *RTT*. As the *RTT* increases the per flow bandwidth**RTT* product increases too, causing *avg* to move away from *maxth* closer to *minth*.

Obviously, the per-flow bandwidth**RTT* product decreases with increasing number of flows. As a consequence, the average queue size moves from a state of oscillation around the minimum threshold to the maximum threshold in case the bandwidth is constant and the number of TCP flows is increased. This has been shown in [4].

In subsequent simulations maxp is set such that the equilibrium point stays roughly at (minth+maxth)/2. RED's minth, maxth and buffersize parameters are set as a function of the bandwidth*RTT product of the specific scenario.

4.2 Constant wq

If *wq* is too high, RED will not filter out transient bursts. An upper bound for *wq* as a function of *minth* and the number of packet arrivals has been given in [1]. If *wq* is too small, RED is not able to detect the initial stages of congestion as the average queue size follows too slowly the changes in the instantaneous queue size. There exists a dependency of wq on RTT and number of flows (see [11]), due to space limitations, however, we solely show the dependency on bottleneck bandwidth.

Simulation3: varying bandwidth, *wq* constant

graph	C	D	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	0.1	100	100	8	33	50	1.3	0.002
b	0.5	100	100	8	33	50	1.8	0.002
c	2	100	100	17	67	100	3.43	0.002
d	50	100	100	100	400	600	160	0.002

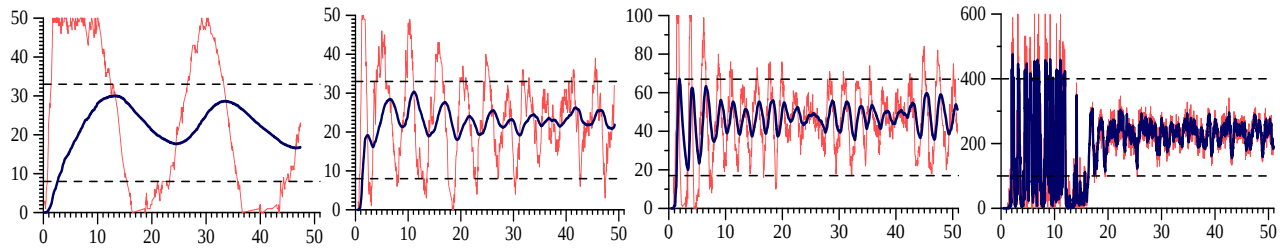

Figure 4 *a-d, inst. and average queue size over time*

In the 100kbps scenario *avg* follows the instantaneous queue size too slowly. This causes convergence of *avg* in between *minth* and *maxth* but severe oscillation of the instantaneous queue size and thereby periods of link underutilization followed by periods of forced drops. Increasing the bottleneck capacity causes the average queue size to track the instantaneous queue size closer, and a decrease in the amplitude of the queue size oscillation (see figures b-d).

4.3 Constant difference between maxth and minth

Simulation4: varying bottleneck bandwidth

graph	C	D	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	0.5	100	100	10	160	213	3.67	0.008
b	5	100	100	10	160	213	7.38	0.002
c	20	100	100	10	160	213	46.5	0.002
d	50	100	100	10	160	213	251	0.002

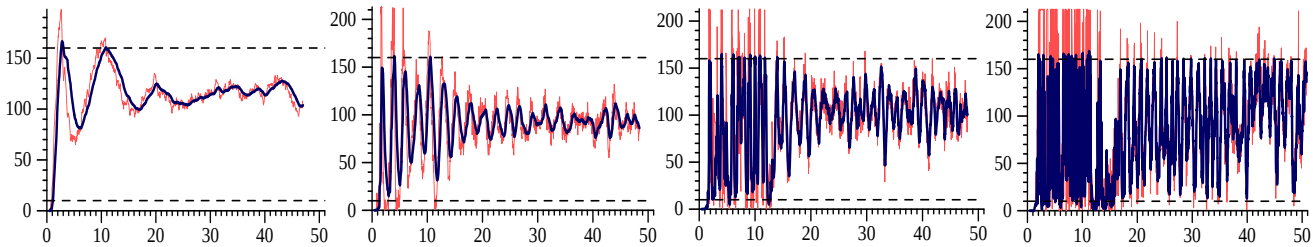

Figure 5 *a-d, inst. and average queue size over time*

Simulation5: varying *RTT*

graph	C	D	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	5	1	100	10	100	133	2.81	0.002
b	5	50	100	10	100	133	4.21	0.002
c	5	100	100	10	100	133	6.31	0.002
d	5	300	100	10	100	133	25	0.002

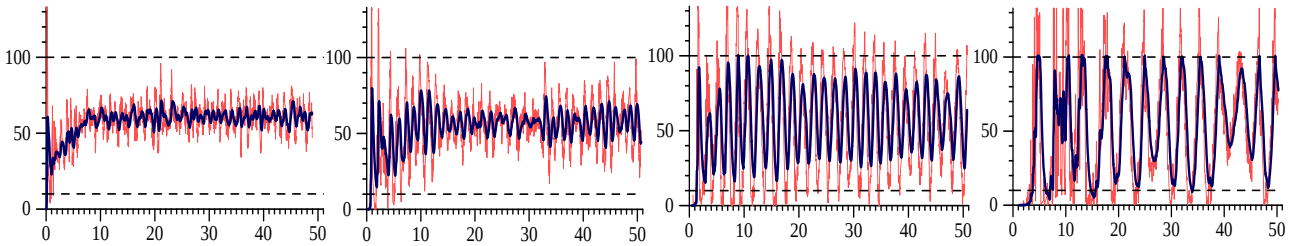

***Figure 6** a-d, inst. and average queue size over time*

As shown in figure 5 and figure 6, the difference between *maxth* and *minth* has to be a monotonically increasing function of the bandwidth*delay product in order to avoid oscillation. If *maxth-minth* is too small compared to the bandwidth*delay product, the queue size oscillates significantly. Simulations in [7] show that the setting of *maxth-minth* to provide convergence of the queue can additionally be considered dependent on the number of flows traversing the link.

5 A Model for RED Parameter Setting

[7] develops a quantitative model how to set RED parameters as a function of the scenario parameters C*RTT and N. This model makes use of a model how to set wq proposed in [11] and provides quantitative rules how to set maxp and maxth-minth in order to achieve convergence of the RED queue. The present paper is based on the model proposed in [7], thus we summarize this model in subsequent paragraphs.

The paper modelling wq [11] finds that setting the RED averaging interval *(I)* equal to the length of the average TCP period of window increase and decrease due to a packet drop, gives a good compromise between the opposing goals of maintaining the moving average close to the long term average and making the moving average respond quickly to a change in traffic conditions (such as an increase in the number of flows). The length of the TCP period *(I)* is derived using results from [13]. For the sampling interval δ an upper bound equal to a minimum RTT is derived. The queue weight can then be computed as

$$wq = 1 - a^{\delta / I} \tag{1}$$

where a is a constant parameter in the order of 0.1.

In order to quantitatively model maxp as a function of C,N and RTT to keep the equilibrium point close to (minth+maxth/2), a steady-state analysis is performed in [7]. This analysis borrows from the fact that the long term average of the sum of the arrival rates of any aggregate of N TCP flows, each flow having a rate R_i, equals the link capacity:

$$L = \sum_{i=1}^{N} R_i \tag{3}$$

Substituting R_i by the expression derived for TCP rates in [13] we get a function of the drop probability p, L, N, and a distribution of round trip times. Our goal is to make the average queue size converge at *(minth+maxth)/2,* which corresponds to a drop probability of *maxp*/2. Thus, substituting p by *maxp*/2, the optimum setting of *maxp* as a function of L,RTT and N can be derived:

$$L = \sum_{i=1}^{N} \frac{1}{RTT_i \sqrt{\frac{bmaxp}{3}} + T_i \min\left(1, 3\sqrt{\frac{3bmaxp}{16}}\right)\left(\frac{maxp}{2} + 4maxp^3\right)} \tag{4}$$

We are not able to provide an analytically derived closed-form expression for *maxp* as solving (3) for *maxp*

results in a polynomial of degree seven. However, numerical solution provided that L,N, and the distributions of RTT and T are given is feasible. For derivation of *maxp* the round trip time RTT_j and the retransmission time-out time T_j of a flow j is set as follows:

$$RTT_j = 2d_j + (minth+maxth)/(2L)$$
$$T_j = RTT_j + 2(minth+maxth)/L \quad (5)$$

The term *(minth+maxth)/(2L)* matches the average queueing delay at the bottleneck, d_j denotes the total propagation delay of RTT class j. As implemented by TCP, T is computed as the RTT plus four times the variance of the RTT, approximated by the average queueing delay at the bottleneck.

Due to the lack of a tractable and sufficiently accurate analytical model [7] uses an empirical approach to determine the setting of *maxth-minth* as a function of the bottleneck bandwidth, *RTT,* and the number of flows. The idea behind this empirical model is that one may find out the proper setting of *maxth-minth* for a specific scenario (as defined by the bandwidth*RTT product and number of flows) by manually adjusting the difference between maxth and minth such that the amplitude of the oscillation of the queue size stays constant at some fraction of maxth-minth. These numerous simulations, exploiting the maximum range of link speeds, propagation delays and number of flows the *ns* simulator allows on high-end PCs, result in a cloud of points in the 3-dimensional (*maxth-minth/bandwidth*RTT/number of flows)* space. The cloud of points is then approximated numerically, yielding a model how to set maxth-minth:

$$maxth - minth = 0.2158 \cdot C \cdot aRTT + 0.567 \cdot N + 84.7 \quad (6)$$

Note that (6) gives a lower bound; simulations show that the amplitude of the oscillation decreases further if *maxth-minth* is set higher than suggested by (6). The term artt denotes the average RTT of a scenario. Equation 6 has been derived based on simulations with 500 bytes packets. [7] additionally shows the constants for (6) based on simulations using other packet sizes.

Above equations are partially interdependent (e.g. artt depends on maxp; maxp depends on minth and maxth) and are thus combined in a non linear system of equations which is solved numerically. Equations 1-6 represent the total quantitative model of RED with bulk-data TCP flows, suitable for heterogeneous RTTs. This model requires as an input the RTT distribution, the number flows and the bottleneck capacity and outputs the RED parameters.

6 Evaluation of the Model

We have executed simulations and measurements with many different input parameters from the (bandwidth*RTT, number of flows) space in [7]. The fundamental result of this analysis is that in all cases the average queue size converges between minth and maxth as desired and the amplitude of oscillation is in the range of (maxth-minth)/2. This paper moves a step further in evaluating the model.

6.1 Web traffic

In addition to the simulations in [7], we investigate the applicability of our RED model to Web-like traffic. For that purpose we use a model of Web traffic which is described in [19]. This model is implemented in recent distributions of ns-2 [17]. Three load scenarios (low, medium, high) are created by changing the number of Web sessions from 600, 700 to 800. The bottleneck bandwidth is set to 5Mbps, bottleneck delay to 100ms; for the rest of the parameter settings we refer to section 3.

Red parameters:

graph	N	minth	maxth	*B*	$maxp^{-1}$	wq
a	50	48	193	290	41	0.001
b	150	69	275	413	9.5	0.0013
c	250	89	356	534	6.2	0.0012

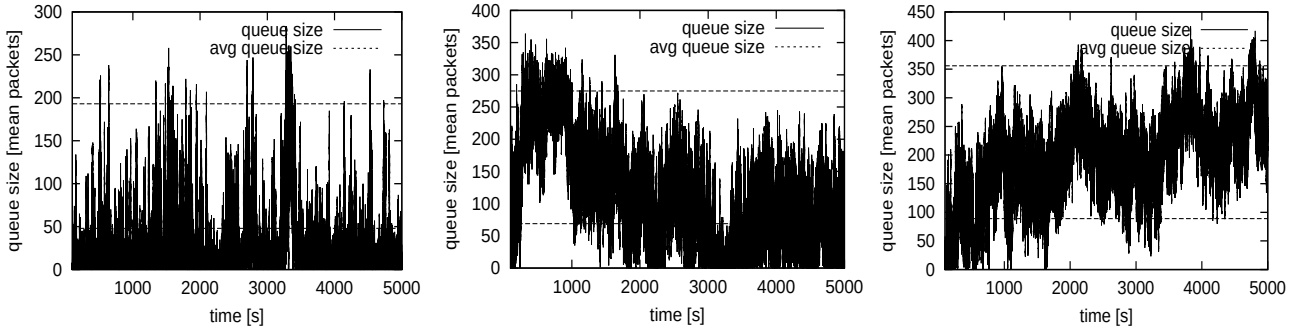

Figure 7 *a-c; RED queue size over time for 600, 700 and 800 Web sessions*

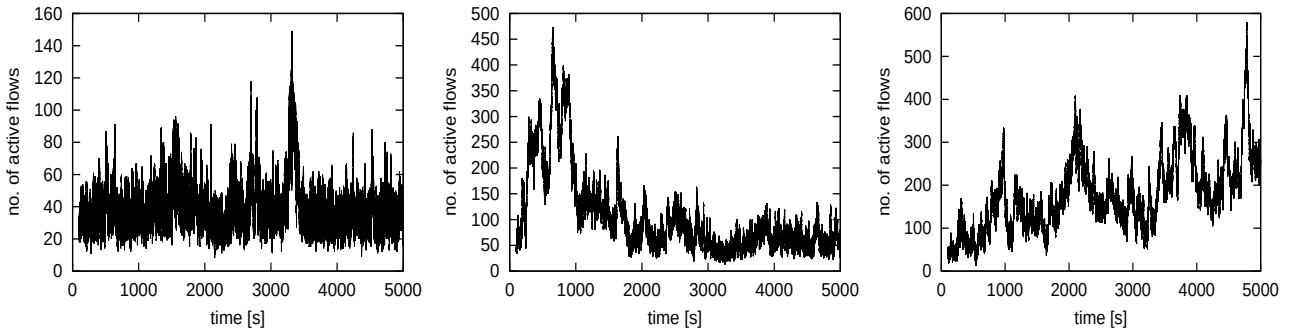

Figure 8 *a-c; Number of active flows over time for 600, 700 and 800 Web sessions*

Figure 7 shows the behavior of the queue size over time. In order to enable the use of our model we need to estimate the number of active flows N (see table above). Unlike the FTP case, the load varies significantly over the course of the simulation as shown by the active number of TCP flows against the time in Figure 8. For the low load scenario the queue size stays low because on average the link capacity is higher than the rate of incoming traffic. For the medium and high load scenarios it can be seen that for some periods of time, the average queue size converges between minth and maxth in a similar way as in the FTP traffic case (see [7]). By comparing the actual load condition and the queue size behavior, it can be seen that these convergence periods go in parallel with periods, where the actual number of flows has approximately the same value as the number of flows parameter that was used as an input to the RED model. This behavior has been observed in all simulations and it is thus concluded that the model for calculating RED parameters is not only suited for FTP traffic but can also be applied to Web-like traffic. It is, however, very difficult (if not impossible) to get a good estimate for the number of concurrent flows as this metric fluctuates heavily. In any case, this is not a weakness of the model (because any model would need a reasonably accurate estimator for the number of flows as an input parameter), but instead a problem of RED's sensitivity to traffic variations.

6.2 End-to-end performance with FTP flows

In this section the performance of the RED algorithm with different parameter settings is compared. One setting is determined through our model. The other settings are such that the buffer size equals the bandwidth*RTT product and the remaining parameters are chosen as given in table 1. The metrics of comparison are aggregate TCP goodput, the standard deviation of the queueing delay a packet infers at the congested router output port, and the number of forced drops from the RED queue management. The simulated topology and traffic generators are the same as documented in section 3; the bottleneck bandwidth equals 10 Mbps, the bottleneck propagation delay equals 100ms.

Table 1 shows the parameter settings for these simulations, covering 10 load scenarios with 20, 60, 40, 140, 180, 220, 240, 260, and 340 flows respectively. In order to keep this table at a reasonable size, however, we only mention 3 load scenarios in table 1 for the simulations based on the model for RED. For other simulations parameter settings stay constant independently of the load.

The parameter set PS 1 is determined through evaluation of the RED model as developed in the previous section. For PS 2, minth is set to 1/4, maxth to 3/4 of the buffer size; the buffersize is set equal to the bandwidth*RTT product; the values for maxp and wq are chosen as recommended in [1]. PS 3 employs a higher constant maxp value and in PS 4 the difference between maxth and minth is chosen smaller than what our model suggests.

Parameter set	PS1 (RED model)	PS1 (RED model)	PS1 (RED model)	PS2 (var. numflows)	PS3 (high maxp)	PS4 (low maxth-minth)
num. flows	20	180	340	20..340	20..340	20..340
minth	50	83	93	83	83	183
maxth	200	331	373	330	330	230
buffersize	301	497	559	500	500	500
1/maxp	665.99	14.14	13.00	10	4	10
wq	0.000166	0.000717	0.000546	0.002	0.002	0.002

Table1: *Parameter settings for simulations*

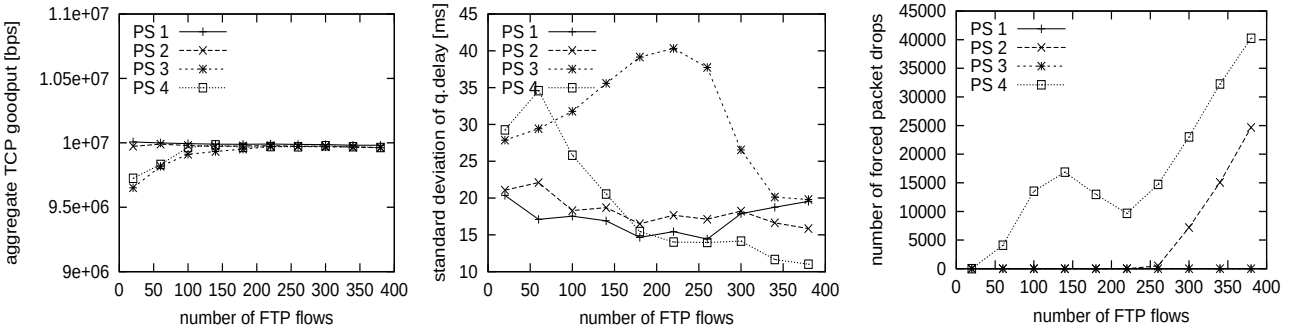

Figure 9 *a-c; TCP goodput, std. dev. of queueing delay and number of forced drops over the number of flows*

As far as the aggregate TCP goodput (figure a) is concerned, the influence of different parameter settings is not very strong. PS 3 and PS 4 perform slightly worse in the low load scenarios due to different reasons. With PS3 the queue size oscillates heavily due to the small difference between maxth and minth and thus the link becomes idle several times (see also section 4.3). With PS4 the high maxp (and thus RED's overly aggressive drop behavior given the specific load situation) enforces TCP flows to back off as soon as the avg. queue-size exceeds minth and thus causes link underutilization (see also section 4.1). Note that goodput would be even worse in case maxp or the bandwidth*RTT product was chosen higher for these scenarios.

The standard deviation of queueing delay within the RED queue is to a high degree influenced by the parameter set employed (see figure b). For PS 1 the deviation is rather constant as our model correctly proposes

a parameter set which establishes low amplitude oscillations of the queue size for all load scenarios. For all other settings, there are some scenarios where the queue size oscillates heavily. With PS 3, this is e.g. the case in medium load scenarios, where the queue size fluctuates between zero and almost maxth. On the other hand, there are scenarios where the standard deviation of queueing delay becomes very low. This is, e.g., the case with PS 2 and PS 4 which can not control the average queue size between minth and maxth but rather make the queue converge to maxth.

Convergence to maxth, however, also results in a high probability of forced packet drops (see figure c). PS 4 is worst in this respect as the improperly small setting of maxth-minth lets the average queue size often exceed maxth. With PS 2, leaving maxp simply constant, RED is unable to control the traffic aggregate in the high load scenarios and thus forced drops are getting more frequent. Recall that forced packet drops are malicious as mechanisms like ECN won't work in this case. With PS1 and PS3 forced packet drops do not occur.

7 Conclusions

This paper aims at systematically investigating the stability criteria of RED and evaluating quantitative guidelines for the setting of RED parameters proposed in companion paper [7]. As the stability of the system does not solely depend on the RED parameters but also on the behavior of end-to-end congestion control and the time scales of the arrival process, we focus on bulk-data TCP traffic for illustration of the dependencies of RED parameter setting on bandwidth*RTT product and number of TCP flows. We show that the queue oscillates heavily if the difference between *minth* and maxth is set too small and that the queue converges to *maxth* or oscillates around *minth* if *maxp* is not set correctly. Subsequently, we briefly summarize the system of equations representing the quantitative model how to set the RED parameters as a function of bottleneck bandwidth, number of flows and round-trip-time distribution and evaluate the model by simulations with realistic Web-like TCP traffic. It is shown that if the input parameters to the model are reasonably accurate (especially the number of active flows parameter), the model provides correct parameter settings.

Subsequently, RED with parameter settings according to the model is compared with RED employing reasonable but non optimized parameter settings. These simulations provide quantitative results on the end-to-end performance decrease in terms of the goodput degradation, high variation in queueing delay and forced packet drops caused by non optimized RED parameter settings.

All in all it is questionable whether appropriate setting of RED parameters is possible in a realistic Internet environment. Although quantitative models like [7] or [14] make an important step ahead in gaining knowledge how to set RED parameters, it is questionable whether ISPs will be able to retrieve sufficiently accurate information about the RTT of flows and the number of flows traversing a router. Additionally, such models can not be applied to scenarios with flows not congestion controlled by TCP without adaptations. This indicates that future queue management algorithms should employ a smaller parameter set and exhibit less dependencies on traffic variability than RED. As shown in [15], "gentle RED" is an example for such an algorithm. Obviously, models for RED parameter setting as proposed in [7] can be applied to any RED derivate.

8 References

[1] S. Floyd, V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993

[2] *B*. Braden, *D*. Clark et al., "Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998

[3] T.J. Ott, T.V. Lakshman, L.H. Wong, "SRED: Stabilized RED", Proc. of IEEE Infocom 1998

[4] W. Feng, D. Kandlur, D. Saha, K.G. Shin, " A self-configuring RED gateway", Proc. of IEEE Infocom 1998

[5] T. Ziegler, U. Hofmann, S. Fdida, "RED+ Gateways for Detection and Discrimination of unresponsive

Flows", Technical Report, November 1998, unpublished
[6] *D*. Clark, "Explicit Allocation of Best Effort Packet Delivery Service", http://www.ietf.org/html.charters/diffserv-charter.html
[7] T. Ziegler, C. Brandauer, S. Fdida, "A quantitative Model for Parameter Setting of RED with TCP Traffic", Proceedings of IWQoS 2001, Karlsruhe, Germany, June 2001
[8] Cisco Web-pages, http://www.cisco.com/warp/public/732/netflow/qos_ds.html
[9] S. Floyd, "Discussions on setting RED parameters", http://www.aciri.org/floyd/red.html , Nov. 1997
[10] C. Villamizar, C. Shong, " High performance TCP in ANSNET", Computer Communication Review, V.24. N.5, October 1994, p. 45-60
[11] V. Firoiu, M. Borden, " A Study of active Queue Management for Congestion Control", Proc. of IEEE Infocom, Tel Aviv, 2000
[12] V. Jacobson, K. Nichols, K. Poduri, "RED in a different Light", Draft Technical Report, Cisco Systems, Sept. 1999
[13] J. Padhye, V. Firoiu, D.Towsley, J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation", Proc. of ACM Sigcomm 1998
[14] C. Hollot et al., "A Control Theoretic Analysis of RED", Proceedings of IEEE Infocom 2001, Tel Aviv, Israel
[15] G. Iannacone, C. Brandauer, T. Ziegler, C. Diot, Serge.Fdida, M. May, "Comparison of Tail Drop and AQM Performance for bulk-data and web-like Internet traffic", to appear in proceedings of ISCC 2001, July 2001, Hammameth, Tunesia
[16] T. Ziegler, S. Fdida, C. Brandauer, B. Hechenleitner, "Stability of RED with two-way TCP Traffic", IEEE ICCCN, Las Vegas, Okt 2000
[17] NS Simulator Homepage, http://www.isi.edu/nsnam/ns
[18] Maple home page, www.maplesoft.com
[19] A. Feldmann, A. Gilbert, P. Huang, W. Willinger, "Dynamics of IP traffic: A Study of the Role of Variability and the Impact on Control", Proceedings of ACM SIGCOMM'99, pages 301-313, 1999