A quantitative Model for the Parameter Setting of RED with TCP Traffic

Thomas Ziegler[1], Christof Brandauer[2], Serge Fdida[3]

[1] FTW, Telecommunications Research Center Vienna, Maderstr.1, 1040 Vienna, Austria
Thomas.Ziegler@ftw.at
[2] Salzburg Research, J. Haringerstr. 5, 5020 Salzburg, Austria
Christof.Brandauer@salzburgresearch.at
[3] Université Pierre et Marie Curie, Laboratoire Paris 6, 75015 Paris, France
Serge.Fdida@lip6.fr

Abstract. This paper systematically derives a quantitative model how to set the parameters of the RED queue management algorithm as a function of the scenario parameters bottleneck bandwidth, round-trip-time, and number of TCP flows. It is shown that proper setting of RED parameters is a necessary condition for stability, i.e. to ensure convergence of the queue size to a desired equilibrium state and to limit oscillation around this equilibrium. The model provides the correct parameter settings, as illustrated by simulations and measurements with FTP and Web-like TCP flows in scenarios with homogeneous and heterogeneous round trip times.

1 Introduction

Although the RED (Random Early Detection) queue management algorithm [1] is already well known and has been identified as an important building block for Internet congestion control [2], parameter setting of RED is still subject to discussion. Simulations investigating the stability of RED with infinite-length (i.e. FTP-like) TCP flows [3][4][5] show that achieving convergence of RED's average queue size (*avg*) between the *minth* and *maxth* thresholds depends on adequate parameter setting. Our observation in [6] is that improper setting of RED parameters may cause the queue size to oscillate heavily around an equilibrium point (the queue average over infinite time intervals) outside the desired range from *minth* to *maxth*. High amplitude oscillations are harmful as they cause periods of link under utilization when the instantaneous queue size equals zero followed by periods of frequent "forced packet-drops" [7] when the average queue size exceeds *maxth* or the instantaneous queue approaches the total buffer size. Forced packet drops are in contradiction to the goal of early congestion detection and decrease the performance of ECN [8], attempting to avoid packet loss and to provide increased throughput for low-demand flows by decoupling congestion notification from dropping packets. In case of WRED [9] or RIO [10] oscillations may cause poor discrimination among in-profile and out-of-profile packets in a DiffServ environment. When the average queue size decreases below the maximum queue size threshold for out-of-profile packets the out-packets may enter the queue. Subsequently, the average queue size increases again and in-packets may be dropped with high probability.

However, the intention of this paper is not to investigate RED's performance decrease due to oscillation around an unfavorable equilibrium point but to build a quanti-

* This work is partly sponsored by the IST project Aquila

tative model for RED parameter setting in case of TCP flows to provide convergence (i.e. desirable behavior) of the queue. Regarding the kind of convergence we may hope to achieve with RED and infinite-length TCP flows it is important to mention that we do not mean convergence to a certain queue size value in the mathematical sense. Contrary, we define convergence very loosely as "achieving a state of bounded oscillation of the queue size around an equilibrium point of (*minth+maxth)/2* so that the amplitude of the oscillation of the average queue size is significantly smaller than the difference between *maxth* and *minth* and the instantaneous queue size remains greater than zero and smaller than the total buffer size".

Realistic Internet traffic consists mainly of short-living (Web-like) TCP flows and not FTP-like traffic, thus it might be stated that a model based on infinite length TCP flows is unrealistic. We argue, however, that a quantitative model for RED parameter setting based on Web-like TCP traffic would be definitely too sophisticated to build. Thus our approach is to build a model for infinite length flows and then investigate if the model provides also proper parameter setting for short living TCP flows. Speaking in other words, our model is based on the commonly used control-theoretic approach of stabilizing a complex, non-linear dynamic system in the steady state case (infinite length flows) as a pre-requirement for desirable behavior in the dynamic-state case (Web flows).

As shown qualitatively in related papers [6][11][12], the RED parameters relevant for stability are the maximum drop probability (*maxp*) determining the equilibrium point, the minimum difference between the queue size thresholds (*maxth-minth*) required to keep the amplitude of the oscillation around the equilibrium point sufficiently low and the queue weight (*wq*). The RED parameters exhibit interdependencies demanding for a systematical approach to derive their values. As a first step, a quantitative model how to set *maxp* dependent on the bottleneck bandwidth, RTT distribution, and the number of flows is derived in section 4. Subsequently, taking into account the *maxp* and *wq* models (see section 2 for *wq*), an empirical model for the setting of *maxth-minth* is derived in section 5. This model gives a lower bound for *maxth-minth* as a function of the bottleneck bandwidth, RTT distribution and number of flows to avoid extensive oscillation of the queue size around the equilibrium point. Finally, after coming back to parameter interdependencies and combining the models for *maxth-minth, maxp* and *wq* into a non-linear system of equations in section 6, simulations and measurements with FTP-like and Web-like TCP traffic are performed to evaluate the model in section 7.

Abbreviations used throughout the paper:
- *B*: buffer size at bottleneck in packets
- *C*: capacity of bottleneck link in Mbps
- *L*: bottleneck-capacity in mean packets per second
- *D*: delay of bottleneck link in ms
- *N*: number of flows

2 Related Research on RED Modelling and Parameter Setting

Existing publications discussing the setting of RED's *minth, maxth* and *maxp* parameters give either rules of thumb and qualitative recommendations [1][3][4][12][13][14] or quantitative models assuming infinite time averages [11] which are not capable of modelling the oscillatory behavior of the RED queue.

[12] proposes a simple quantitative model how to set *wq* based on RED's response to a unit-step input signal. In [11] the length of the TCP period of window increase and decrease *(I)* and an upper bound of one RTT for RED's queue-sampling interval (δ) is derived using the TCP model proposed in [15]. It is found that setting the RED averaging interval equal to *I* results in 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. RED's averaging interval equals the TCP period *I* if the queue weight is computed as follows:

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

where *a* is a constant parameter in the order of 0.1, providing a quantitative model for the setting of *wq*.

Independently and simultaneously to the present paper (see [6] for an earlier version) a control theoretic approach to model the parameter setting of RED has been made in [16]. This paper is based on a linearization of the TCP model published in [17].

Additionally, [18] compares the end-to-end performance of RED variants and Drop-Tail. [19] shows that RED-like mechanisms tend to oscillate in the presence of two-way TCP traffic. [20] investigates the tuning of RED parameters to minimize flow-transfer times in the presence of Web traffic.

3 Simulation Settings

Simulations are performed with the *ns* network simulator [21] using the topology shown in figure 1. All 500Mbps links employ Drop Tail queue management; buffer sizes at access links are set sufficiently high to avoid packet loss. Thus packets are discarded solely at the bottleneck link from router1 to router2.

The link between router1 and router2 uses RED queue management. RED is operated in packet mode only as simulations in [6] and [12] show the same results 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.

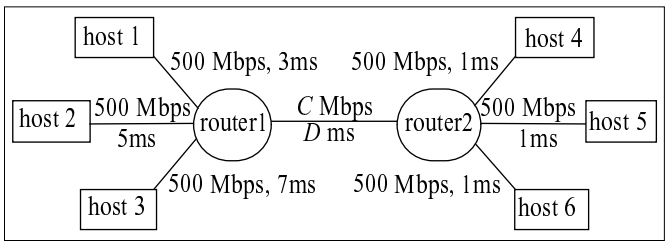

Fig. 1. Simulated network

Unless otherwise noted, *N* TCP flows start at random times between 0 and 10 seconds of simulation time. Hosts at the left hand side of the simulated network act as sources, hosts at the right hand side act as sinks. Host *i* starts *N/3* TCP flows to host *3+i*. TCP data senders are of type Reno and New-Reno. Packet sizes are uniformly distributed with a mean of 500 bytes and a variance of 250 bytes.

In all queue size over time figures shown in this paper the average queue size is plotted with a bold line, the instantaneous queue size is plotted with a thin line. The unit of

the x-axis is seconds, the unit of the y-axis is mean packet sizes. Simulations last for 100 seconds. However, only the last 30 seconds are plotted in queue size over time figures as we are rather interested in the steady-state than the startup behavior of the system.

4 Determining the Equilibrium Point

The following simulation shows that *maxp* has to be set as a function of the aggressiveness of traffic sources in order to enable convergence of the average queue to an equilibrium point between *minth* and *maxth*. The aggressiveness of an aggregate of TCP flows is inversely proportional to its per-flow bandwidth*RTT product[1], defined as *C*RTT/N*. Thus we can for instance increase the bottleneck capacity *C* and leave *maxp* constant to illustrate the drift of the equilibrium point.

Constant parameters: *D* = 100ms, *N* = 100; *maxp* is set to 0.1 as recommended in [13]. Lacking models for *maxth, minth* and *wq* these parameters have to be set reasonably based on experience from former simulations and rules of thumb for simulations in this chapter. The buffer size *B* is set to the bandwidth*RTT product of the scenario or 40 packets, whichever is higher; *maxth = 2B/3, minth = maxth/4*. These settings result in a difference between *maxth* and *minth* sufficiently high to avoid high amplitude oscillations, and a setting of *wq* to make the average queue size track the instantaneous queue properly as evaluated in earlier simulations and shown also in figure 2.

Simulation	C	minth	maxth	B	wq
1	2	29	114	171	0.0042
2	10	143	571	857	0.00063
3	50	714	2857	4286	0.000035

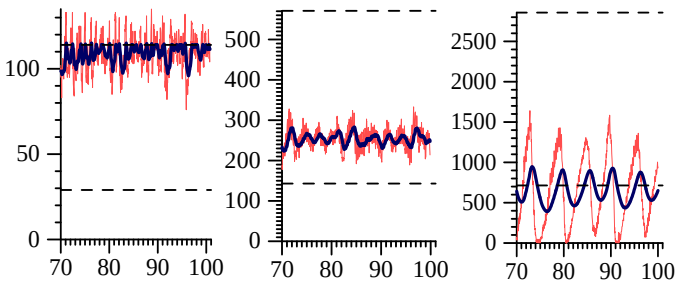

Fig. 2. Simulations 1-3, inst. and average queue size over time, maxp = 1/10

As the increase of *C* causes the per-flow bandwidth*RTT product to increase, the aggressiveness of the TCP flows and thereby the average queue size decreases. In simulation 1 a constant *maxp* of 0.1 (and thus the RED drop probability) is too small, causing convergence of *avg* to *maxth* and thus frequent forced packet drops. In simulation 2 *maxp* is well chosen given the specific scenario, thus the queue's equilibrium point is in-between *minth* and *maxth*. In simulation 3 a *maxp* of 0.1 is too high causing *avg* to oscillate around *minth* resulting in suboptimal link utilization as the queue is often empty.

---

[1] The per-flow bandwidth*RTT product can also be considered as the long term average TCP window of flows.

4.1 A model for maxp

We present a steady-state analysis resulting in a quantitative model on how to set *maxp* as a function of *C*, *N* and *RTT*. As shown in [15], the rate R_i of a TCP flow *i* in units of pkt/s as a function of the loss probability (*p)*, the round trip time (RTT_i) and retransmission time-out time (T_i) can be modelled by the following expression:

$$R_i = \frac{1}{RTT_i \cdot \sqrt{\frac{2bp}{3}} + T_i \cdot min\left(1, 3 \cdot \sqrt{\frac{3bp}{8}}\right) p(1 + 32p^2)} \tag{1}$$

The constant *b* in the above equation denotes the number of packets received at the TCP data receiver to generate one ACK. With a delayed ACK TCP data receiver *b* would be equal to 2. For the moment, we do not use delayed ACKs, hence *b* is set to one.

For any aggregate of *N* TCP flows the long term average rate equals the link capacity:

$$L = \sum_{i=1}^{N} R_i \tag{2}$$

Substituting R_i by eq. 1 we get a function of *p*, *L*, *N,* a distribution of round trip times and retransmission time-out times. Our goal is to make the average queue size converge at *(minth+maxth)/2,* which corresponds to a drop probability of *maxp/2*. Substituting *p* with *maxp/2*, we can derive the optimum setting of *maxp* as a function of *L*, *RTT* and *N*:

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

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*, the distributions of *RTT* and *T* are given is feasible with a mathematics software like [22].

ISPs are not aware of the exact distribution of RTTs of flows passing a RED queue. The best we may hope to obtain in order to provide a practical model is a histogram, grouping flows into *m* RTT classes where each RTT class *j* has n_j flows and homogeneous round trip times. For such a model, eq. (3) can be rewritten as

$$L = \sum_{j=1}^{m} \frac{n_j}{RTT_j \cdot \sqrt{\frac{bmaxp}{3}} + T_j min\left(1, 3\sqrt{\frac{3bmaxp}{16}}\right)\left(\frac{maxp}{2} + 4maxp^3\right)} \tag{4}$$

For derivation of *maxp* in subsequent simulations we have set RTT_j and T_j as follows:

$$\begin{aligned} RTT_j &= 2d_j + (minth+maxth)/(2L) \\ T_j &= RTT_j + 2(minth+maxth)/L \end{aligned} \tag{5}$$

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.

The following paragraphs repeat the simulations at the beginning of this section but with *maxp* adapted according to the model:

Simulation	4	5	6
1/maxp	4.15	30	618

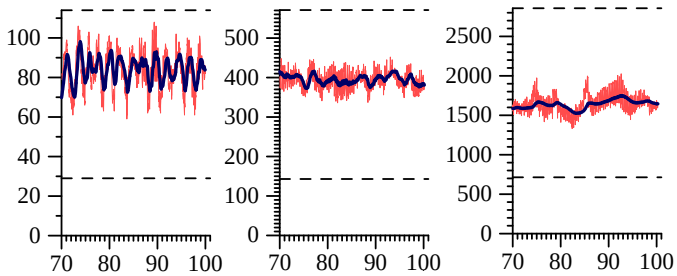

Fig. 3. Simulations 4-6, inst. and average queue size over time, maxp adapted

Figure 3 shows that adapting *maxp* according to equation 4 makes the queue size converge close to *(minth+maxth)/2*.

For the remainder of the paper, *maxp* is set according to the model proposed in this section. Having a model for *maxp* we may from now on additionally use the model how to set the *wq* parameter as explained in [11] and section 2. Note that the model for the setting of *wq* parameter is based on eq. 1 and thus requires the drop probability and RTT as an input. As for the derivation for the *maxp* model we may approximate the drop probability for the *wq* model as *maxp/2*; for the RTT we refer to equation 5. As the *ns* RED implementation computes the average queue size at each packet arrival the δ parameter is set to the inverse of the link capacity in packets. The constant a is set to 0.01 instead of 0.1 (see section 2) resulting in a slightly higher queue weight parameter and thus marginally shorter memory in RED's queue averaging compared to [11]. Simulations show evidence that choosing parameters for the setting of *wq* as described above results in the average queue size properly tracking the instantaneous queue size and avoids possible oscillations by choosing *wq* too small (i.e. too long memory).

5 Determining the Amplitude of the Queue Size Oscillation

As TCP is volume controlled, the buffer requirements of an aggregate of TCP flows solely depends on the product of bandwidth and RTT, no matter how the individual values for bandwidth or RTT are set in a scenario. Simulations 7-9 successively increase D to vary the bandwidth*RTT product, showing the dependency of *maxth-minth* on bandwidth and RTT. Constant parameters: C = 20Mbps, *minth* = 40, *maxth* = 300, B = 400. N is adapted such that the per flow bandwidth*RTT product stays roughly constant.

Simulation	D	N	1/maxp	wq
7	1ms	18	42	0.0025
8	50ms	67	37.5	0.0015
9	300ms	317	36	0.00017

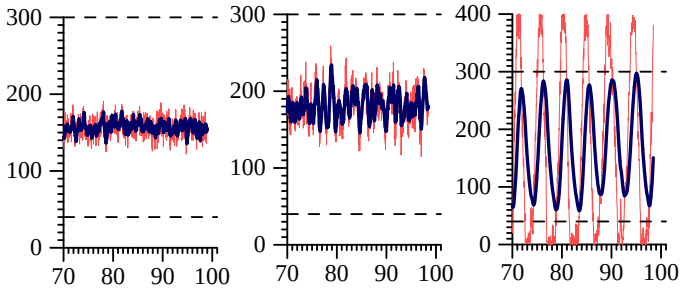

Fig. 4. Simulations 7-9, inst. and average queue size over time

The difference between *maxth* and *minth* has to be a monotonically increasing function of the bandwidth*RTT product in order to bound the oscillation of the average queue size. In case the difference between *maxth* and *minth* is left constant and the bandwidth*RTT product is increased (see figure 4), the amplitude of the oscillation may be improperly high (simulation 9) or even unnecessarily low (simulation 7).

Simulations 10-12 investigate the behavior of the RED queue in case of a varying number of TCP flows. Constant parameters: C = 20Mbps, D = 100ms, *minth* = 40, *maxth* = 300, B = 400.

Simulation	N	1/maxp	wq
10	100	49	0.0004
11	250	10.3	0.00065
12	300	8	0.0007

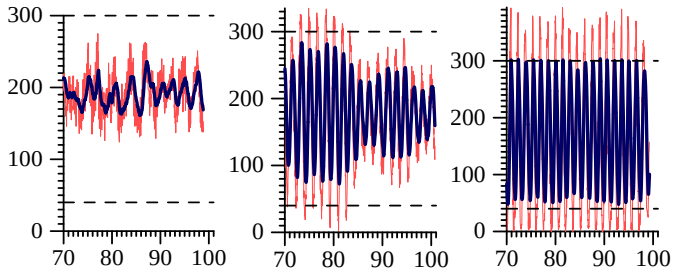

Fig. 5. Simulations 10-12, inst. and average queue size over time

According to our model for *maxp*, an increase in the number of TCP flows requires setting *maxp* higher in order to keep the equilibrium point close to *(minth+maxth)/2* (see parameter table for figure 5). Increasing *maxp* and leaving *maxth-minth* constant means increasing the slope of RED's drop probability function as defined by *maxp/(maxth-minth)*. A steeper drop probability function means that small changes in TCP congestion windows, in other words small changes in the queue size, cause high changes in RED's drop probability. In response to a high change in drop probability, TCP flows drastically alter their congestion windows causing the oscillatory behavior shown in figure 5, simulation 12. Thus, to avoid these oscillations, we conclude that an increase of *maxp* (due to an increase in N) must be accompanied by an increase of *maxth-minth* in order to bound the slope of the drop probability function.

5.1 Model for maxth-minth and homogeneous RTTs.

Employing the qualitative insights from the introduction of this section and the models already derived for *maxp* and *wq* to make the queue converge to *(maxth+minth)/2* we use an empirical approach to quantitatively model the required difference between *maxth* and *minth* as a function of the bottleneck capacity and round-trip-time product *(C*RTT),* and the number of flows *(N)*. For the moment we only consider the case of homogeneous RTTs (section 5.2 generalizes the model to the heterogeneous RTT case).

The goal of our model is to determine the difference between *maxth* and *minth* such that the amplitude of the oscillation around the equilibrium point is kept constant at a desirable value of *(maxth-minth)/4.*[2] This should happen independently of the *C*RTT* and *N* input parameters. Our approach towards generation of such a model is to perform simulations over a broad range of input parameters and thereby manually adjust *maxth-minth* until the above condition is met. The resulting cloud of points in the 3-dimensional *(C*RTT, N, maxth-minth)* space can then be approximated numerically by a closed-form expression, providing the desired quantitative model for *maxth-minth*.

In all simulations the total buffersize (*B*) is set to 3**maxth*/2, sufficiently high to avoid losses due to buffer overflow; *minth* equals *(maxth-minth)/3;* packet sizes have a mean of 500 bytes, see section 3. For other simulation settings we refer to section 3.

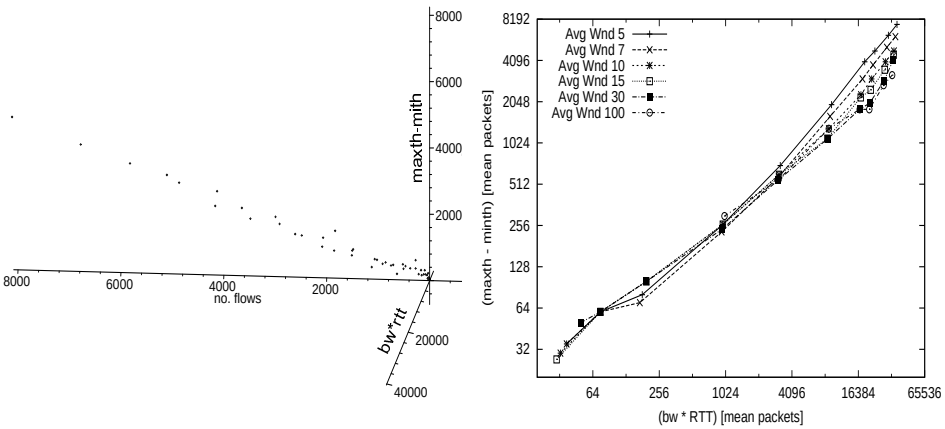

Fig. 6. Simulation results for maxth-minth (in mean packets) as a function of the bandwidth*RTT product (in mean packets) and the number of flows

Figure 6a shows the resulting cloud of points where each point represents a proper value for *maxth-minth* which has been found by simulation using our empirical approach. Figure 6b illustrates how figure 6a has been created and ameliorates its readability. Six sets of simulations have been conducted for creation of figure 6a, each consisting of 10 simulations having identical average per-flow windows (as defined by *C*RTT/N*) and being represented by one curve in figure 6b.

The distribution of points in figure 6a and the curves in figure 6 show that *maxth-minth* can be considered as almost linearly dependent on the bandwidth*RTT product

---

[2] The choice of *(maxth-minth)/4* for the oscillation of the average queue size reflects a reasonable compromise between having stability margins for a bounded oscillation and trying to keep queueing delay and buffer requirements as small as possible.

and the number of flows. Thus the cloud of points in figure 6a may be approximated by a linear least-square fit, yielding *maxth-minth* as a linear function in two variables:

$$maxth - minth = c1 \cdot C \cdot RTT + c2 \cdot N + c3, \quad (6)$$

with the values of *c1 = 0.02158*, *c2 = 0.567*, *c3 = 85* for an average packet size of 500 bytes. We have repeated our empirical approach for packet sizes of 250, 1000, and 1500 bytes and found analogous approximations as different packet sizes did not change the linear shape of the cloud of points; see table 1 for the resulting constants.

avg. packet size	c1	c2	c3
250	0.02739	0.7324	17
500	0.02158	0.5670	85
1000	0.01450	0.3416	46
1500	0.01165	0.09493	85

Table 1: Constants for *maxth-minth* model

Note that eq. 6 gives a lower bound; simulations show that the amplitude of the oscillation decreases further if *maxth-minth* is set higher than suggested by eq. 6. Although eq. 6 results in settings of *maxth-minth* significantly smaller than the bandwidth*delay product (compare to [12]), the buffer requirements of RED with TCP traffic are definitely substantial in high speed WANs. For instance, considering 50000 FTP flows over a network path with a propagation delay of 100ms and a 5Gbps bottleneck, the required difference between *maxth* and *minth* equals 54000, and the buffer size equals 107000 500 byte packets according to the model. The bandwidth*RTT product for this scenario equals 250000 500 byte packets.

We have exploited the maximum range of link speeds and number of flows the *ns* simulator allows on high-end PCs for the derivation of the model for *maxth-minth*, including scenarios with more than 6000 TCP flows and 200Mbps link speed. We argue that 6000 flows over a 200Mbps link are likely to behave similar to for instance hundred thousands of flows over a Gigabit link, giving reason to believe that our model can be extrapolated to higher link speeds and higher number of flows than possible with ns. Thus the empirical model for *maxth-minth* should not yield drawbacks concerning its applicability to a wide spectrum of scenarios compared to an analytical approach. Additionally, common simplifications in analytical papers are not required when performing simulations.

5.2 Incorporating heterogeneous RTTs

Equation 6 provides a model how to set *maxth-minth* as a function of the bottleneck bandwidth, number of flows and the RTT, assuming that all flows have homogeneous (i.e. equal) RTTs. In order to extend the model to the heterogeneous RTT case, we propose to compute the RTT quantity in eq. 6 as a weighted average (*aRTT*) of the per-flow RTTs. A TCP flow's share of the bottleneck capacity and the bottleneck queue is inversely proportional to its RTT. Flows utilizing a higher portion of the bottleneck resources have greater influence on the queue dynamics than flows utilizing a smaller portion of the bottleneck resources. It is thus reasonable to use a flow's share of the link capacity for computation of the weights for *aRTT*.

Similar to section 4.1, we assume a histogram of *m* RTT classes, where each RTT

class j has n_j flows and homogeneous round trip time RTT_j. The rate R_j of a TCP flow joining class j can be computed according to eq. 1, section 4.1. Again, we approximate the drop probability required for the TCP model as *maxp/2*. The average RTT, weighted by the rate of flow aggregates can be computed as follows:

$$aRTT = \sum_{j=1}^{m} \frac{n_j R_j}{L} RTT_j \tag{7}$$

Finally, equation 6 has to be rewritten as

$$maxth - minth = c1 \cdot C \cdot aRTT + c2 \cdot N + c3 \tag{8}$$

6 Parameter Dependencies and Model Assembly

RED parameter settings are partially interdependent. Thus it is important to take these dependencies into account when designing a model. The *maxp* parameter determines the equilibrium point of the queue size and depends on *C, N* and *RTT. RTT* means that *maxp* depends on queueing delay and thus on the setting of *maxth* and *minth*. However, the equilibrium point (as it is the *infinite* time average), does not depend on how accurately RED's average queue size follows the instantaneous queue size. Thus *maxp* can be derived independently of the setting of *wq* (compare section 4.1).

The *wq* parameter, on the other hand, depends on the number of flows, the bottleneck bandwidth, the drop probability (and thus on *maxp*) and the RTT (and thus implicitly on *maxth* and *minth*). Thus we can compute the *wq* parameter having the model for *maxp* and assuming fixed settings for *maxth* and *minth*.

The setting of *maxth-minth* depends on *maxp, wq, RTT*, the bottleneck bandwidth and the number of flows. The dependency on *maxp* and *wq* is taken into account by our empirical model for *maxth-minth* as the *maxp* and *wq* models are used as an input.

Deriving the model for *maxth-minth* yields an equation for *maxth-minth* as a function of *C*RTT* and *N*. Although *maxp* and *wq* have been taken into account for the derivation of the model, they do not appear directly in the equation (compare to eq. 6) due to the empirical approach of deriving it. Thus, assuming homogeneous RTTs, we may first compute *maxth-minth* and then compute *maxp* knowing how to set *maxth* and *minth* when applying the model to compute the RED parameters for scenario defined by *C*RTT* and *N*. Finally *wq* can be computed knowing *maxp, maxth* and *minth*.

For the following reason it is not possible anymore to consider the computation of *maxp* and *maxth-minth* as independent from each other in the case of heterogeneous RTTs: As explained above the setting of *maxp* depends on *maxth* and *minth*. As shown in section 5.2, the setting of the *aRTT* quantity and thus the setting of *maxth-minth* depends on *maxp*. As a consequence, equation (8) and equation (4) have to be combined to a system of non-linear equations which can be solved numerically for *maxp, minth,* and *maxth*. Finally, knowing how to set *maxp, minth,* and *maxth* for computation of the average length of the TCP period *I*, *wq* can be computed as explained in section 2.

The quantitative model for RED parameter setting has been implemented in the *maple* mathematics software [22] and can be easily used via the Web, see [23].

7 Evaluation of the Model

7.1 FTP-like Traffic

More than 200 simulations with arbitrary points in the (C*RTT,N) space, with the TCP delayed ACKs option enabled and disabled and with homogeneous and heterogeneous RTTs have been conducted to evaluate the model for FTP-like TCP flows (see also [6]). In all these simulations the queue size converges between *minth* and *maxth* as desired. This paper only shows a small subset of these simulations as the new insights the reader may gain are restricted.

Simulation	C	D	N	minth	maxth	B	1/maxp	wq
13	1	0.01	3	31	125	187	280	0.002540
14	50	0.1	313	176	703	1054	35	0.000172
15	150	0.25	2163	1064	4255	6383	36	2.383e-05

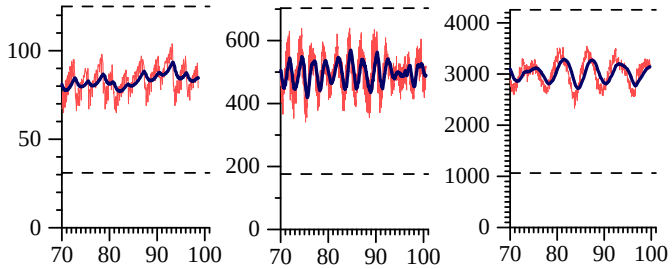

Fig. 7. Simulations 13-15, inst. and average queue size over time

Figure 7 shows that the parameter settings proposed by the model are appropriate for the selected scenarios.

A model partially developed empirically by simulation should not only be evaluated by simulation. Thus we have performed extensive (more than 50) measurements with various points in the (C*RTT,N) space using a topology as illustrated in figure 8.

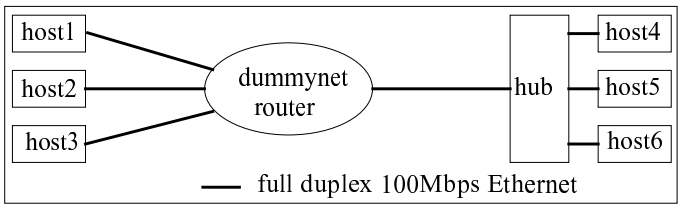

Fig. 8. Measured network

The propagation delay of all links can be assumed to be equal zero as all equipment is located in one lab; the packet processing time in nodes has been determined as negligible. TCP data senders are located at host1-3, TCP data-receivers at hosts 4-6. All hosts are Pentium III PCs, 750Mhz, 128 MB RAM and run the ttcp tool to generate the FTP-like bulk-data TCP flows. Host operating system is Linux running kernel version 2.2.16 implementing TCP SACK with an initial congestion window of two segments and delayed ACKs. The initial congestion window property of the Linux TCP implementation makes sources somewhat more aggressive compared to the simulations.

The "dummynet router" is another Pentium III PC, 750Mhz, 128MB RAM running the FreeBSD operating system version 3.5. This PC runs the dummynet tool [24] for emulation of a bottleneck link (specified by bandwidth *C*, delay *D* and an MTU of 500 bytes) and the RED queue management algorithm. Note that dummynet operates only on packets flowing in direction host1-3 to host4-6. In the reverse direction dummynet is transparent. Congestion and packet drops happen solely due to RED at dummynet's emulated bottleneck link. The bottleneck capacity is chosen sufficiently small to make the influence of collisions at the Ethernet layer on the measurement results negligible. Additionally, we have verified carefully that all PCs provide sufficient computational power in order to avoid influencing the measurement results by side-effects.

Measurement	C	D	N	minth	maxth	B	1/maxp	wq
1	0.5	50	10	33	132	197	69	0.003
2	3	100	200	76	303	455	5.9	0.00159
3	10	50	50	48	193	290	41.3	0.000966
4	10	300	200	118	471	706	57	0.000178

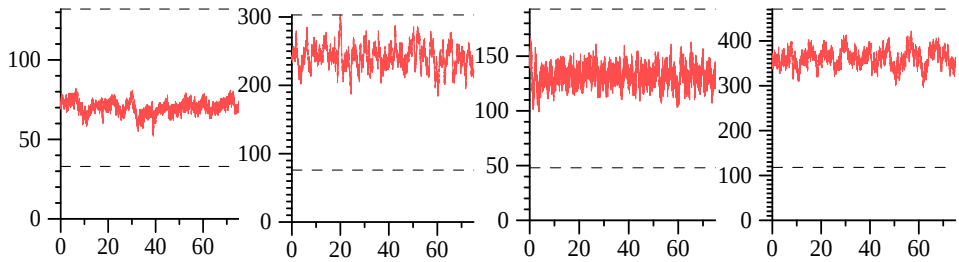

Fig. 9. Measurements 1-4, RED's instantaneous queue size over time

Figure 9 has been created by dummynet dumping the instantaneous queue size on the harddisk every time a packet arrives at the RED queue. As illustrated in figure 9, the measurements confirm simulations in verifying the correctness of the model. Measurements 2 and 4 exhibits a somewhat higher equilibrium point, which can be explained by the TCP model (equation 1) becoming less accurate for very high drop probabilities (this effect has also been observed in simulations) and the aggressive kind of TCP implemented in the used Linux OS (initial congestion window equals 2 segments).

7.2 Web Traffic

This section simulates Web-like TCP SACK traffic, employing a model for HTTP 1.0 as described in [25] and being shipped with *ns-2*. We simulate *U* Web client-server connections. A client downloads a page consisting of several objects from the server, is idle for some think time and downloads the next page. Between getting two subsequent objects, the client waits for some inter-object time. According to [25] all random variables of the model are Pareto distributed. Table 2 shows the parameters of the distributions.

parameter of pareto distribution	think time	objects per page	inter object time	object size
mean	50 ms	4	0.5 ms	12 Kbyte
shape	2	1.2	1.5	1.2

Table 2: Parameters for Web model

The *N* parameter (number of active TCP flows) required as an input to the RED parameter model is estimated by the mean number of active flows over the entire simulation time. Comparing the number of Web-sessions and the average number of flows in the subsequent table, we can observe a non linear dependency of traffic load (as indicated by *N*) on the number of Web Sessions *U*.

Constant parameters: C = 10Mbps, D = 0.1s, packet size = 500 bytes.

Simulation	N	minth	maxth	B	wq	1/maxp	U
16	50	56	224	336	0.000371	113	900
17	250	97	388	581	0.000748	9.7	1500
18	1000	250	999	1498	0.000478	4.2	1600

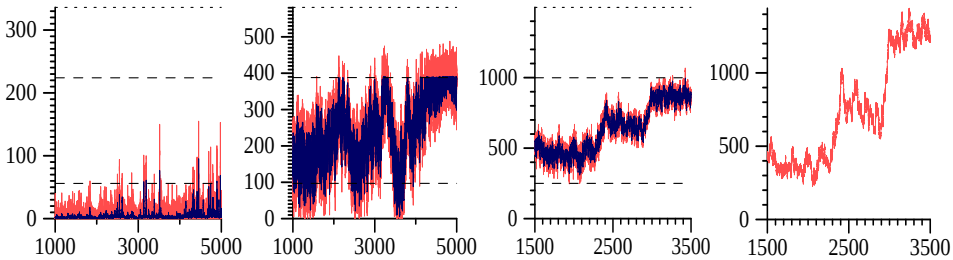

Fig. 10. Simulations 16-18, inst. and average queue size over time. Simulation 18, number of active flows over time

Simulations 16 to 18 show scenarios with low, medium and high load due to different Web traffic demands. For all scenarios, the queue size curves depend to a great extent on the traffic demand created by the Web-sessions which is highly variable over time. This is illustrated in the two rightmost part-figures of figure 10, showing the queue size and the number of active flows (an indicator for the instantaneous traffic demand) over time for simulation 18. The queue size curve tracks the number of active flows curve closely.

In the lightly loaded scenario (simulation 16) the queue hardly exceeds *minth* due to the low traffic demand, thus RED has no influence on system dynamics. Simulations 17 and 18 represent scenarios with sufficient load to fill the buffer dependent on the variations in instantaneous traffic demands. Of course we can not expect convergence of the queue with bounded oscillation between *minth* and *maxth* similar to the case of FTP-like flows for such a scenario. However, the RED model proposes reasonable parameter settings as the RED queue buffers most traffic bursts between *minth* and *maxth*.

A comparison of the number of flows and the queue size over time curve for simulation 18 explains why the queue size may deviate extensively from the desired value of *(minth+maxth)/2* and proofs that this deviation happens due to the variability in load and not due to an inaccuracy of the RED model. Up to a simulation time of about 2400 seconds the real number of active flows is smaller than 1000 (the value of the *N* parameter used as input to the model) thus the queue size is below the desired value. After 2800 seconds the real number of flows exceeds *N* making the queue approach *maxth*. From 2400 to 3000 seconds of simulation time, however, the real number of active flows is approximately equal to *N* and the queue size is close to the desired value. Thus we can conclude that the model provides proper parameter settings as long as we are able to estimate the input parameters to the model with sufficient accuracy (see section 8 for a remark). This has been observed also in other simulations with Web traffic.

8 Conclusions

We find that the RED parameters relevant for stability are the maximum drop probability (*maxp*) determining the equilibrium point (i.e. the queue average over infinite time intervals), the minimum difference between the queue size thresholds (*maxth-minth*) required to keep the amplitude of the oscillation around the equilibrium point sufficiently low and the queue weight (*wq*). The major part of this paper is dedicated to the systematical derivation of a quantitative model for the setting of the RED parameters mentioned above to achieve stability of the queue size. The model is suitable for heterogeneous RTTs and assumes TCP traffic. The stability of the system does not solely depend on the dynamics of the RED drop-function and end-to-end congestion control but also on the time scales of the arrival-process. It would be, however, too complex to build a quantitative model with realistic Web-like TCP traffic, thus we focus on infinite life-time, bulk-data TCP flows for derivation of the model.

Note that there exist other constraints for RED parameter setting which are not considered in this paper as they are not relevant for system stability but which are nevertheless important (e.g. setting *minth* as a trade-off between queueing delay and burst tolerance). For a discussion of these constraints we refer to [13].

The model is evaluated by measurements with bulk-data TCP traffic and simulations with bulk-data and Web-like TCP traffic, showing that RED parameters are set properly and that deriving the RED parameter model under the simpler steady state conditions (infinite length flows) provides good parameter settings not only for the steady state case but also for the dynamic state case (realistic Web-like flows). Web traffic, however, has a highly variable demand thus it is difficult to determine a good, fixed value for an estimator of the traffic load (average number of active flows) which is required as an input parameter to *any* model proposing parameter settings for active queue management mechanisms. As a consequence, we can not expect the queue to stay always within some desired range even though the model for parameter setting itself may be correct. Showing the correctness of our model for Web traffic, we have demonstrated by simulation that if we are able to feed our RED model with sufficiently accurate input parameters, stability (i.e. bounded oscillation of the average queue size between *minth* and *maxth*) can be achieved for Web traffic. In a realistic environment, however, it is questionable whether ISPs are capable of retrieving sufficiently accurate information about the RTT of flows and the average number of flows traversing a router. Additionally, the number of flows is highly variable thus finding a good fixed value for e.g. the number of flows may not be possible. Simple extensions to RED may help diminishing dependencies on the scenario parameters and thus provide performance benefits compared to the original version of RED. For instance, gentle RED decreases the dependency on traffic variability in terms of the number of flows due to its smoother drop function, as shown in [18]. However, no matter whether the gentle or the original version of RED is used, the model proposed in this paper can provide useful insights how to set active queue management parameters.

Acknowledgements

The authors are grateful to Eduard Hasenleithner, Ferdinand Graf and Matthäus Gruber from Salzburg University of Applied Sciences for building the testbed and supporting the measurements.

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. W. Feng, D. Kandlur, D. Saha, K.G. Shin, “ A self-configuring RED gateway”, Proc. of IEEE Infocom 1998
4. W. Feng et al., “Techniques for eliminating Packet Loss in congested TCP/IP Networks”, University of Michigan CSE-TR-349-97, November 1997
5. T.J. Ott, T.V. Lakshman, L.H. Wong, “SRED: Stabilized RED”, Proc. of IEEE Infocom 1998
6. T. Ziegler, S.Fdida, C. Brandauer, “Stability Criteria of RED with TCP Traffic”, Tech. Report, unpublished, August 1999, http://www-rp.lip6.fr/publications/production.html
7. S. Floyd, K. Fall, K. Tieu, “Estimating arrival rates from the RED packet drop history”, March 1998, unpublished, http://www.aciri.org/floyd/end2end-paper.html
8. K.K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC2491, January 1999
9. Cisco Web pages, http://www.cisco.com/warp/public/732/netflow/qos_ds.html
10. D. Clark, “Explicit Allocation of Best Effort Packet Delivery Service”, http://www.ietf.org/html.charters/diffserv-charter.html
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. S. Floyd, “Discussions on setting RED parameters”, http://www.aciri.org/floyd/red.html , Nov. 1997
14. C. Villamizar, C. Shong, “ High performance TCP in ANSNET”, Computer Communication Review, V.24. N.5, October 1994, pp. 45-60
15. J. Padhye et al.,”Modeling TCP Troughput: A simple Model and its empirical Validation”, Proceedings of ACM SIGCOMM, August 1998
16. C. Hollot et al., “A Control Theoretic Analysis of RED”, To appear in proceedings of IEEE Infocom 2001, Tel Aviv, Israel
17. V. Misra, W. Gong, D. Towsley, A Fluid-based Analysis of a Network of AQM Routers Supporting TCP Flows with an Application to RED”, SIGCOMM 2000
18. 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
19. T. Ziegler, S. Fdida, C. Brandauer, B. Hechenleitner, "Stability of RED with two-way TCP Traffic", IEEE ICCCN, Las Vegas, Oct. 2000
20. Mikkel Christiansen, Kevin Jeffay, David Ott, F. Donelson Smith, “Tuning RED for Web traffic”, Proc. of ACM SIGCOMM 2000
21. NS Simulator Homepage, http://www.isi.edu/nsnam/ns/
22. Maple home page, http://www.mit.edu/afs/athena.mit.edu/software/maple/www/home.html
23. RED model homepage, www.salzburgresearch.at/~cbrand/REDmodel
24. L. Rizzo, “An embeddded Network Simulator to support Network Protocols’ Development”, Proceedings of Tools’97, St. Malo, France, June 1997
25. 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