University of Würzburg
Institute of Computer Science
Research Report Series

Analytic Performance Evaluation of the RED Algorithm for QoS in TCP/IP Networks

Stefan Köhler, Michael Menth and Norbert Vicari

Report No. 259 March 2000

Department of Distributed Systems
Institute of Computer Science
University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
Phone: +49 931 888 6644 Fax: +49 931 888 6632
koehler|menth|vicari@informatik.uni-wuerzburg.de

Analytic Performance Evaluation of the RED Algorithm for QoS in TCP/IP Networks

Stefan Köhler, Michael Menth and Norbert Vicari
Department of Distributed Systems
Institute of Computer Science
University of Würzburg
Am Hubland, D-97074 Würzburg, Germany
Phone: +49 931 888 6644 Fax: +49 931 888 6632
koehler|menth|vicari@informatik.uni-wuerzburg.de

Abstract

The Random Early Detection algorithm is considered as a promising algorithm for Differentiated Services to cope with feedback oriented traffic like TCP. In this paper, we model the salient features of TCP and RED ad evaluate the model with a discrete-time analysis. The novelty of our investigation is the analytic approach to the composed model, i.e., TCP feedback traffic over a RED queue. Deriving not only mean values but also distributions for the performance measures, we obtain insights of the behaviour of TCP under RED.

Keywords: RED, TCP/IP, DiffServ, Discrete-Time Analysis

1 Introduction

The TCP congestion avoidance mechanisms [1] are a means to deal with congestion situations in a growing Internet. Without an active buffer management the queues run full or exhibit the 'Lock-Out' phenomenon [2]. While high buffer occupation entails loss of packet bursts, that are characteristic for TCP/IP traffic, the 'Lock-Out' phenomenon provides unfairness between different TCP connections.

The Random Early Detection (RED) [3] algorithm is currently the most promising and intensively discussed algorithm to further enhance TCP/IP connection performance and is recommended as one possible solution for these issues. It is also proposed as an active queue management algorithm for the future Differentiated Services scenario [4]. Since separate queues are provided for different service classes, it is even more important to minimize long-term congestion while allowing short-term congestion resulting from bursts. This is achieved by providing gradual congestion feedback within each traffic class, which is obtained by a properly designed RED queue.

Even though RED is known since 1993 [3] and has attracted lot of attention in the last two years, the optimal dimensioning of RED queue parameters is not fully understood, yet. Most of the existing studies are obtained with simulations suggesting modifications to the algorithm. For example, FRED [5] proposes the use of per flow information to enhance the performance of RED gateways. The number of active connections is taken into account in the RED flavors SRED [6] and Self-Configuring RED [7]. Throughput and

fairness are the most frequent addressed criteria, while the interaction of RED parameters and TCP properties are not considered in these studies.

Recently, some effort was undertaken to evaluate the performance of RED analytically. In [8, 9] the dependence of RED performance with respect to bursty traffic is studied. A first approach to investigate RED in the presence of feedback traffic is presented in [10] where a source consists of a 3-state Markov model. In our study, the sources behave as defined in the TCP Reno protocol.

We set up an analytical model for a TCP connection and validate it by simulations. The combination with RED yields an analytical model for TCP data transmission over a RED queue. Distributions for various performance measures allow insights to the TCP behavior influenced by RED.

The paper is organized as follows: In the next section we introduce the investigated TCP version and explain the functionality of RED. In Section 3 the mathematical discrete-time model is developed and numerical results are discussed in Section 4. Section 5 concludes the work and gives an outlook for further studies.

2 Modeling TCP and RED

In the following section TCP and RED are explained to an extend for understanding the modeling of section 3.

2.1 Introduction to TCP

TCP is a reliable connection-oriented, end-to-end protocol designed to protect a network against heavy overload and to achieve a fair share of the bandwidth for different TCP sources. TCP detects errors like damaged, lost or duplicated packets. The sender assigns a sequence number to every variable-length packet (segment) and requires a positive acknowledgment (ACK) from the receiver. The receiver detects packet loss by checking the sequence number and acknowledges the last packet received in correct order every time a packet is received. Consequently, for every packet which is not in order a duplicate acknowledgement is sent. The purpose of this duplicate ACK is to let the peer know that a loss occurred and to tell the sender the next expected sequence number.

We consider a saturated TCP source like in a FTP session where all segments are of the maximum possible size resulting into constant-length packets. The term packet is used equivalent to the term segment. We refer to TCP Reno with mechanisms from RFC 2001 and 2581 [1, 11] and model it similarly to [12]. TCP has a congestion window and a slow start threshold which are described by the variables CWND and SSTHRESH. CWND determines the maximum number of packets that are allowed to be unacknowledged in the network and SSTHRESH to control the growth of CWND. If CWND $<$ SSTHRESH holds (cf. Figure 1), that is in the slow start phase, the congestion window is increased by one packet for every received non duplicate acknowledgement. This results in an exponential growth of CWND. When CWND is greater than or equal SSTHRESH, i.e., in the congestion avoidance phase, CWND is growing by one packet per round trip time (RTT) which corresponds to a linear growth of CWND (see Figure 1).

TCP reacts to packet loss in two different ways. It sets a timer for each sent packet. If a timeout happens before the acknowledgement returns for that packet, CWND is reduced to

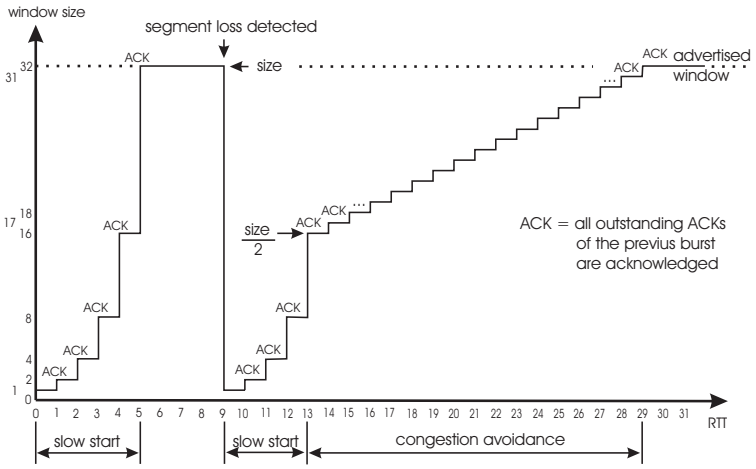

Figure 1: Simplified illustration of the TCP's slow start and congestion avoidance algorithm.

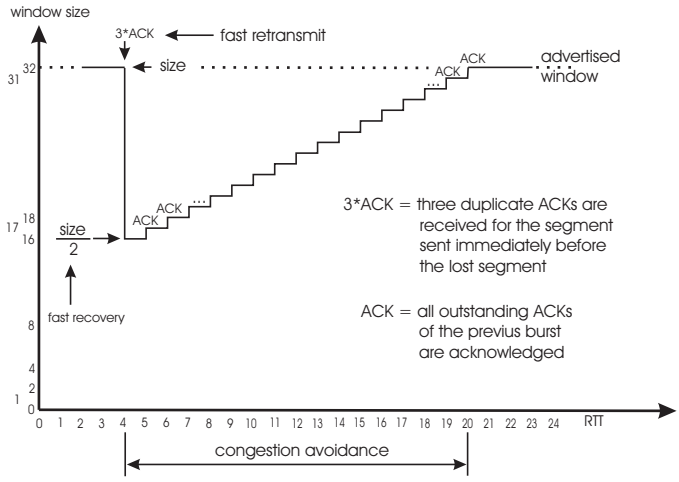

Figure 2: Simplified illustration of the TCP's fast retransmit algorithm.

the maximum segment size and SSTHRESH is set to $max(\text{flight size}/2, 2 * \text{segment size})$ (see Figure 1). Flight size denotes the amount of unacknowledged data in the network. Since we assume a saturated sender the value of CWND is equal to value of the flight size. When the sender receives three duplicate ACKs before a time out occurs, a fast recovery is done: In a simplified manner, TCP sends the lost packet again, reduces SSTRHESH to $max(\text{flight size}/2, 2 * \text{segment size})$ and sets CWND to the new calculated SSTHRESH (see Figure 2).

We assume for our model that the propagation delay dominates all other delays. No acknowledgment are lost, packet losses happen independently of each other and more than one lost packet induces a TCP timeout.

With these assumptions the TCP model is comparable to a back-to-back batch model. The batch size is determined by the current congestion window size. If the batch arrives at the receiver, a batch of acknowledgements is generated and sent back immediately. The sender receives them after two propagation delays which corresponds to an RTT. Now, the sender continues, hence, we can talk about TCP rounds. This behaviour is often found in

TCP connections with a long RTT.

2.2 Random Early Discard (RED) Queue

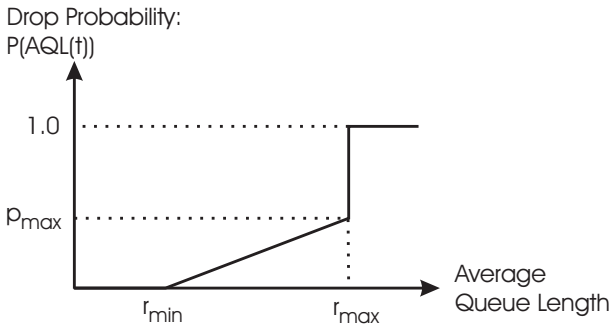

Figure 3: Increasing drop probability in RED.

RED was initially proposed by Floyd and Van Jacobson [3]. A simple drop tail queue is extended by a counter A for the average queue size. The average queue size A_n is calculated at the arrival of the n-th packet: the present queue size Q is considered by the weight factor w_q and the old average queue size A_{n-1} by $1 - w_q$ [3]. The intention of this averaging process is to react only to persistent overload situations and not to burst arrivals. The RED queue defines a loss probability $p(A = i)$ for the arriving packets depending on the average queue occupation. The loss function p can be parameterized as follows:

$$p(A = i) = \begin{cases} 0 & 0 \leq i < r_{min} \\ (\frac{i - r_{min}}{r_{max} - r_{min}}) \cdot p_{max} & r_{min} \leq i \leq r_{max} \\ 1 & r_{max} < i \leq \infty \end{cases} \quad (1)$$

Arriving packets are lost with the probability of this defined loss function when the average queue size exceeds a certain threshold r_{min}.

When carrying TCP traffic of multiple connections over a RED queue, packets are discarded early and not prevalently in situation of queue overflow. It is likely that several randomly chosen connection experience a single packet loss instead of one connection suffering for a multiple packet loss. Therefore, a sufficient number of connections receive feedback from a heavy loaded network in time and adapt their transmission rates instead a few connections breaking down to their lower limit. This is intended to improve the fairness between TCP connections.

3 Analysis of TCP Data Transmission over a RED Queue

In this section, an analysis for data transmission using TCP over a RED queue is derived. The proceeding is like in [13]: a discrete-time Markov model of the system is set up to find its stationary state distribution. Based on this, several performance measures are computed.

For the sake of simplicity some notations are introduced regarding an arbitrary random variable X:

$X_{min}, \quad X_{max}$	minimum and maximum value of X,
$\mathcal{X}$	range of values for X,
$X(Y = i)$	random variable X conditioned on $Y = i$,
$x, \quad x[i]$	distribution of X and probability $\Pr(X = i)$,
$x(Y = i)$	distribution conditioned on $Y = i$,
X_n	random variable X at the n-th observation point,
$\overline{X} = \sum_{i=X_{min}}^{X_{max}} x[i] \cdot i$	mean of X,
$c_{var}(X) = \frac{\sqrt{\sum_{i=X_{min}}^{X_{max}} x[i] \cdot (i-\overline{X})^2}}{\overline{X}}$	coefficient of variation of X.

3.1 Discrete-Time Models

In the next subsection, we explain the mathematical model of a single TCP connection, then the pure RED queue mechanism and, finally, the behavior of several TCP connections over a RED queue. The model is observed at discrete observation instants $t_n \in \mathbb{N}_n$. A model comprises a set of variables describing the present state of the model that contains all the necessary information to determine the state at the next observation point t_{n+1}, given a certain random event which is denoted as the model factor.

3.1.1 A Discrete-Time Model for a Single TCP Connection

We consider the TCP mechanisms after each round, since we are interested in the batch size that a saturated source emits under different conditions for packet loss. The end of a round corresponds to a discrete observation point $t_n \in \mathbb{N}_0$. The state of the TCP connection is described by its congestion window size $CWND$, here denoted by W, and its slow start threshold $SSTHRESH$, here denoted by S. The behavior of the TCP model depends on the number of losses $L(W)$ that occur during a round. The factor is represented by a conditioned random variable that depends on the number of unacknowledged packets W and is binomially distributed since packet losses are considered to happen independently of each other:

$$l(W = j)[i] \quad = \quad \binom{j}{i} p^i \cdot (1-p)^{j-i}. \tag{2}$$

The algorithm works reactively, therefore, the adaptation by TCP is delayed by one round, i.e., the number of losses in a round must be remembered by the random variable M (memory). The state of the TCP connection and the last loss variable M compose the model state at the observation point t_n, denoted by the triple (W_n, S_n, M_n). The model state and the number of losses in the following round L suffices to predict the model state at the next observation point t_{n+1}.

The state transition is done by Algorithm 1. If no loss occurred during the last round, the slow start threshold S is not changed. If the maximum window size is reached, the window size is not changed, either. In case of slow start, the window size is doubled or in the congestion avoidance phase increased by one packet per round. If loss occurred within the last round, the SSTHRESH is reset to half the present window size. If exactly one loss occurred, the window size is set to the resulting SSTHRESH, otherwise it is torn

down to 1. Finally, the number of losses occurred in this round must be recorded for the next one. Note that it does not matter whether 2 or more packets are lost, therefore, either 0, 1 or 2 losses are remembered.

Note that $W_n - L(W_n)$ is the effectively transmitted batch size in a round.

Input: model state (W_n, S_n, M_n), model factor $(L(W_n))$
  if $(M_n = 0)$ **then** {no loss last round}
    $S_{n+1} := S_n$
    if $(W_n = W_{max})$ **then** {full window possible}
      $W_{n+1} := W_n$
    else
      if $(W_n < S_n)$ **then** {slow start}
        $W_{n+1} := 2 \cdot W_n$
      else {congestion avoidance}
        $W_{n+1} := W_n + 1$
      end if
    end if
  else
    if $(M_n = 1)$ **then** {one loss last round}
      $S_{n+1} := \max(W_n/2, 2)$
      $W_{n+1} := S_{n+1}$
    else {more than one loss last round}
      $S_{n+1} := \max(W_n/2, 2)$
      $W_{n+1} := 1$
    end if
  end if
  $M_{n+1} := \min(L(W_n), 2)$
Output: model state $(W_{n+1}, S_{n+1}, M_{n+1})$

Algorithm 1: TCP - A Single TCP Connection.

3.1.2 A Discrete-Time Model for a Server Implementing RED

We assume a server with constant bandwidth C implementing the RED mechanism. First, the queue is observed at an arbitrary time instant t_0. The queue length at observation point t_n is denoted by Q_n. The next observation point is determined by $t_{n+1} = t_n + \max(1, \frac{Q_n}{C})$, i. e., when all packets in the queue at t_n are transmitted (cf. Figure 4). We also talk about a RED round.

The amount of data that has arrived between two observation points is given by the independently and identically distributed random variable B. We assume that $B > 0$ holds which is true in case of saturated TCP sources. The number of occurred losses given by $L(B, A)$ and depends on the number of arrived packets B and the average queue size A since it controls the loss. The difference $Q = B - L(B, A)$ yields the present queue size. The average queue size A_{n+1} is updated by weighting the new queue size Q and the old average A_n by a weighting factor w_q. Updating A at the observation points is an approximation of the reality in contrast to an averaging process based on the packet

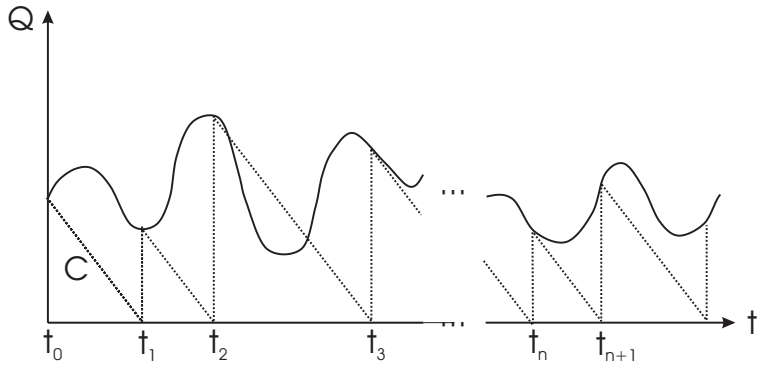

Figure 4: Renewal Points.

arrival instants. Hence, in this case, a higher w_q must be chosen to realize a comparable memory effect.Algorithm 2 describes the model behaviour.

Input: model state (A_n), model factor $(B, L(B, A_n))$
$Q := B - L(B, A_n)$
$A_{n+1} := \lfloor w_q \cdot Q + (1 - w_q) \cdot A_n \rfloor$
Output: model state (A_{n+1})

Algorithm 2: RED- The RED Queue Mechanism.

The probability of k losses within a batch of j packets follows a binomial distribution since each packet sees the same loss probability $p(A)$ (Equation 1):

$$l(A = i, B = j)[k] \quad = \quad \binom{j}{k} p(A = i)^k \cdot (1 - p(A = i))^{j-k}. \tag{3}$$

3.1.3 A Discrete-Time Model of TCP Connections over a RED Queue

The last step is to bring the TCP and the RED model together. We assume h TCP connections transmitting data over a RED queue. The model state is described by the random variables (Wi, Si, Mi), $i \in \{1, ..., h\}$ and (A). The loss that a connection encounters is determined by $Li(Wi, A)$ as outlined in Equation 3. All the TCP sources are saturated, therefore, they are waiting for $\sum_{i=1}^{h} Wi$ acknowledgements, i. e., the number of packets at the server $Q = \sum_{i=1}^{h} \big(Wi - Li(Wi, A)\big)$ is the difference of the window sizes Wi and the lost packets $Li(Wi, A)$. To build a discrete model, the queue can be observed at the same instants as in Section 3.1.2 because within this time, for each TCP source exactly one round completes and the behavior of the TCP sources does not differ from Section 3.1.1 except for the round alignment which has no influence on the window sizes at t_n. Hence, the basic functionality of the TCP and RED mechanism does not change. The coupling consists of propagating the loss induced by the loss function $p(A)$ of the RED queue as feedback to the TCP connections to control their window size.

The traffic arriving at the RED queue is the sum of all congestion window sizes $\sum_{i=1}^{h} Wi$ representing the number of sent packets during the next round of each connection. The number of lost packets within the next round is the sum of lost packets of all TCP connections $\sum_{i=1}^{h} Li(A, Wi)$.

Given this, the compound model is characterized by Algorithm 3 which relies on Algorithm 1 and Algorithm 2.

Input: model states (Wi_n, Si_n, Mi_n), $i \in \{1, ..., h\}$, $(A_n))$, model factors $Li(A_n, Wi_n)$ $i \in \{1, ..., h\}$
for $i \in \{1, ..., h\}$ **do**
$(Wi_{n+1}, Si_{n+1}, Mi_{n+1}) :=$ **TCP**$((Wi_n, Si_n, Mi_n), Li(A_n, Wi_n))$
end for
$(A_{n+1}) :=$ **RED**$((A_n), (\sum_{i=1}^{h} Wi_n, \sum_{i=1}^{h} Li(A_n, Wi))$
Output: model state $((Wi_{n+1}, Si_{n+1}, Mi_{n+1}))$, $i \in \{1, ..., h\}$, $(Q_{n+1}, A_{n+1}))$

Algorithm 3: TCPoverRED - h TCP Connections under RED.

3.2 Stationary Distribution

To derive the stationary state distribution of the model a numerical framework for solving discrete and finite Markov models [14] is applied which is basically a generalized formalization of the method used in [15]. The framework is extended by the use of two-dimensional conditional distributions, the generation of a start vector by a short Markov chain simulation, and an accelerated iteration scheme.

Only a description of the Markov model is needed from which the numerical program can be syntactically deduced. The description comprises renewal points, state variables, factors influencing the system and a state transition function describing the behavior of the system.

- The renewal points of the Markov model are the observation points just before the end of a RED round.
- The state variable $X = ((Wi, Si, Mi), i \in \{1, ..., h\}, (A))$ and
- the factor variable $Y = (Li(A, Wi), i \in \{1, ..., h\})$ are according to Section 3.1.3.
- The state transition function f, that describes the state evolution between two renewal points, is given by Algorithm 3.

To find the stationary (average) state distribution x, the state distributions x_n are needed. Starting with an initial probability vector x_0, the successor distribution x_{n+1} is computed using f, x_n and the distribution y of the factor.

$$x_{n+1}[i] = \sum_{\{(j,k) \in \mathcal{X} \times \mathcal{Y}(j) | f(j,k)=i\}} x_n[j] \cdot y[k] \tag{4}$$

The limit of their average eventually yields the stationary state distribution x.

$$x = \lim_{k \to \infty} \frac{1}{k} \sum_{n=0}^{k-1} x_n \tag{5}$$

For the single TCP connection model and for the simple RED queue model, as outlined in Section 3.1.1 and Section 3.1.2, the stationary state distribution is found in the same

way. However, these distributions take not into account the time between the observation points. A state $X = ((Wi, Si, Mi)i \in \{1...h\}, (A))$ lasts $D(X) = \frac{1}{C} \cdot \sum_{i=1}^{h} \big(Wi - Li(Wi, A)\big)$ time. The state distribution over time xt is computed using the average state duration $\overline{D(i)}$:

$$\overline{D(X,Y)} = \frac{1}{|\mathcal{X}|} \cdot \sum_{i \in \mathcal{X}} \cdot \sum_{j \in \mathcal{Y}} D(i,j) \tag{6}$$

$$xt[i] = \frac{x[i] \cdot \sum_{j \in \mathcal{Y}} D(i,j)}{\overline{D(X,Y)}} \tag{7}$$

3.3 Performance Measures

To compute the distribution of a random variable U, which stands for any one of W, S, M, A, Wi, Si and Mi, we define the set $\mathcal{U}(i) = \{j \in \mathcal{X} \mid U = i\}$ of all states that fulfill the condition $U = i$. Then the probability of $U = i$ is simply obtained by summing up the probabilities of the states in $\mathcal{U}(i)$:

$$u[i] = \sum_{j \in \mathcal{U}(i)} x[i] \text{ or} \tag{8}$$

$$ut[i] = \sum_{j \in \mathcal{U}(i)} xt[i] \text{ , respectively.} \tag{9}$$

To get the distribution of U over time ut, xt must be employed.

In doing so, the distribution for W in Section 3.1.1 is found as well as the queue occupation Q and the congestion window sizes Wi in Section 3.1.3.

The distribution of the transmitted packets $Ti = Wi - Li(Wi, A)$, which is the difference of the congestion window size and the number of lost packets, can be obtained in a similar way.

$$ti[j] = \sum_{k=0}^{Li_{max}} \sum_{r \in \mathcal{W}i(j+k)} x[r] \cdot li(r)[k]. \tag{10}$$

Since the model in Section 3.1.3 is symmetrically designed, the distribution of the window size Wi and the number of transmitted packets Ti is identical for all connections.

4 Numerical Results

In this section the TCP model is validated comparing results from analysis and simulation. Then the effects of the loss function and weighting factor of a RED queue on TCP connections are studied.

4.1 Results for a Single TCP Connection

We consider a large TCP connection with fixed packet loss probability. Since Microsofts operating system Windows uses a TCP buffer of 8760 Bytes and the maximum Ethernet TCP segment size is 1460 Bytes, the maximum value for CWND is 8760/1460 = 6. Measurements have proven this parameter in more than 70% of TCP connections [16].

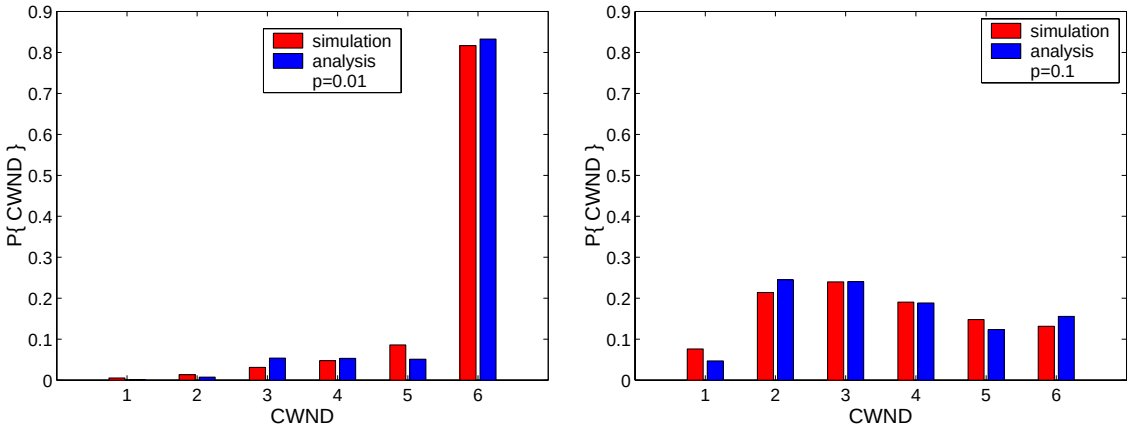

Figure 5: Comparison of analytical and simulated TCP congestion window distribution.

Figure 5 presents a comparison of the analytically derived distribution with simulation results generated with the ns simulator [17]. The distribution of the congestion window CWND for drop probabilities $p = 0.1$ and $p = 0.01$ are found in good accordance with the simulated results.

4.2 TCP Sources over a RED Queue

In the following numerical results, that relate to TCP data transmission over a RED queue, are presented. If not stated differently, the results stem from a RED queue, where the parameters were set to: $r_{min} = 9$, $r_{max} = 18$ packets and $w_q = 0.5$. The RED queue is shared by 3 TCP sources, that greedily generate packets according to a maximum congestion window CWND $= 6$.

4.2.1 Influence of the moving average of the Queue Occupancy

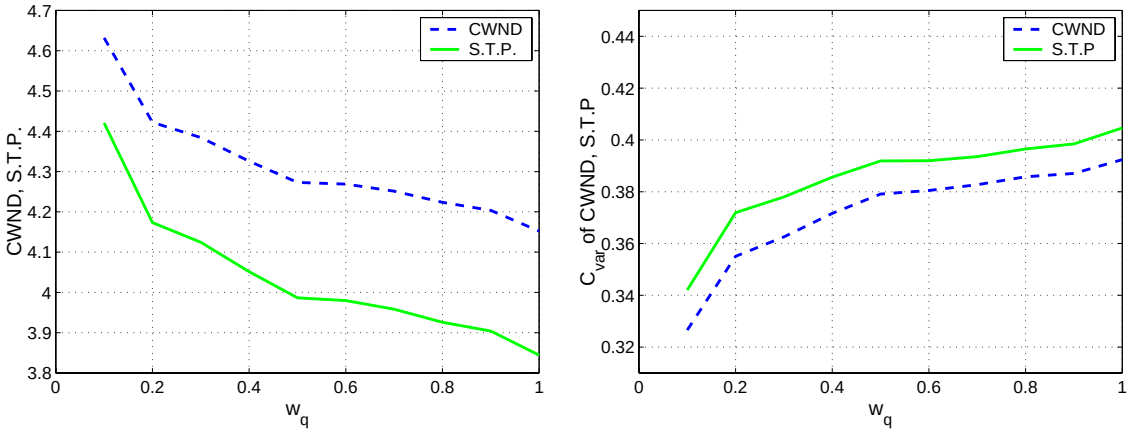

Figure 6: Smoothing of TCP bursts by averaging the queue occupancy.

The computation of the average queue occupancy is considered an important factor for the dimensioning of RED. Figure 6 (left) depicts the mean congestion window size

and the number of successfully transmitted packets in dependence of the weighting factor w_q. A drop probability $p_{max} = 0.5$ is applied. Both performance indicators are reduced when increasing w_q, that is, taking into account only recent values of queue occupancy and thus not smoothing bursts. Discarding packets from large bursts more aggressively increases the variance of the congestion window size (cf. Figure 6) which is a indication that the TCP control returns more often to the initial state and, thus, reduces throughput.

4.2.2 Influence of the Loss Function

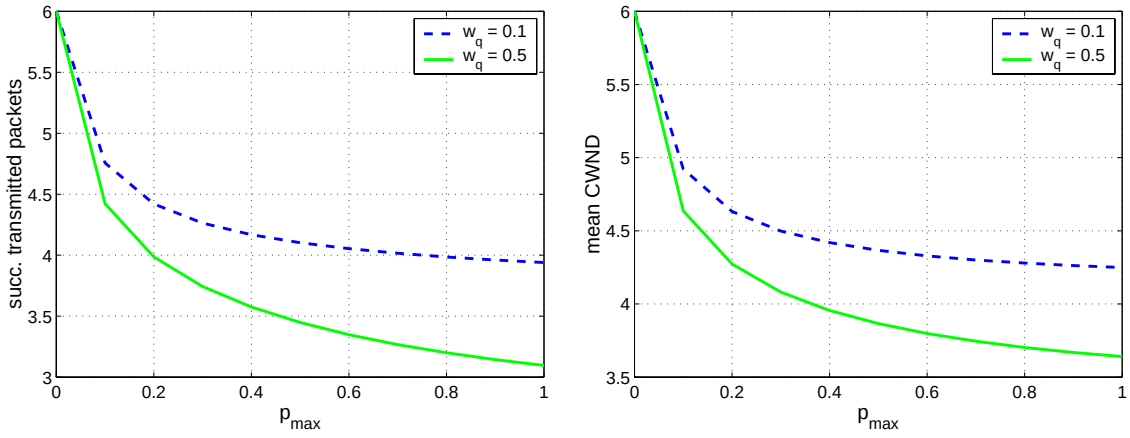

Figure 7: Increasing the slope of the loss function reduces system performance.

In Figure 7 the basic working mode of RED is demonstrated. The left graph shows that increasing the slope $\frac{p_{max}}{(r_{max}-r_{min})}$ of the loss function, the RED algorithm causes packet losses and thereby reduces the number of successfully transmitted packets. Induced by the packet loss the average congestion window, depicted in Figure 7 (right), is reduced according to the TCP mechanism.

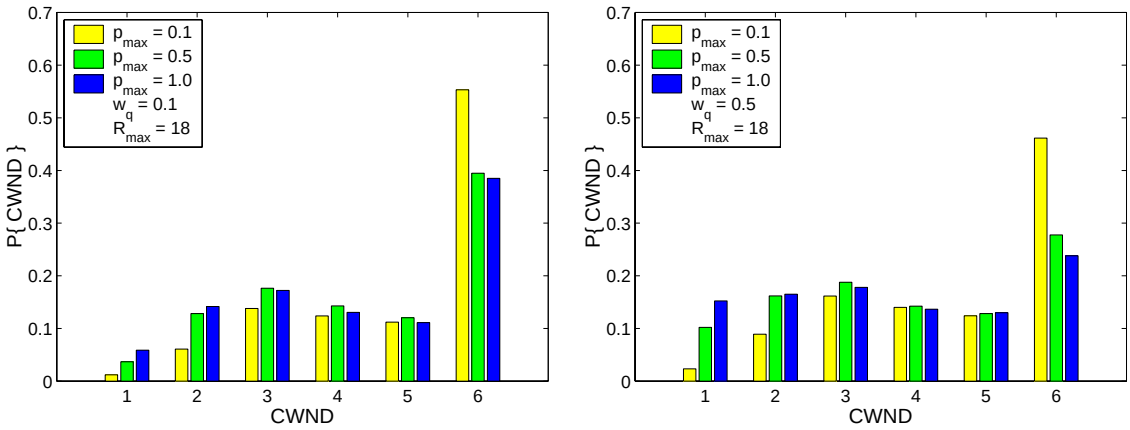

Figure 8: Distribution of CWND for $w_q = 0.1$ and $w_q = 0.5$.

For high values of the drop probability p_{max} the CWND size stabilizes or slightly decreases. This is explained by the distribution of CWND as shown in Figure 8 for $w_q =$

0.1 and $w_q = 0.5$. Increasing the loss probability from 0.1 to 0.5 reduces in both situations the probability to be in the maximum state. The behavior is lost when p_{max} is further increased. For the system with longer memory, that is $w_q = 0.1$, the average queue length reflects lower TCP states for a longer time and thus enables TCP to grow to the maximum state.

4.2.3 Influence of the Buffer Size

The virtual buffer size determined by the loss function parameter r_{max} is of relevance for proper system dimensioning. We choose a fixed w_q of 0.3 and investigate its influence by reducing the maximum average queue size from 18 to 12.

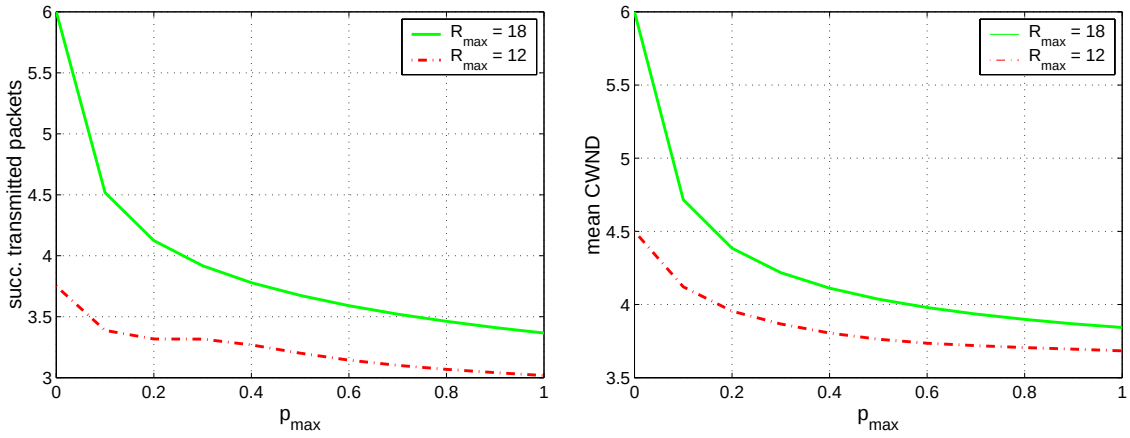

Figure 9: RED performance with reduced average queue size.

For a drop probability $p_{max} = 0$ the system behaves like a smoothed drop tail queue and transports for the well designed system ($r_{max} = 18$) all offered traffic. For the reduced queue ($r_{max} = 12$) the resources are shared fair. As observed in Figure 9 (left) the performance for a smoothed drop tail queue is always better than the RED queue. The number of successfully transmitted packets for the reduced queue attains a local minimum at $p_{max} = 0.2$ and a local maximum at $p_{max} = 0.3$. The size of the congestion window depicted in Figure 9 (right) shows that TCP reacts on the losses caused by r_{max} and low p_{max} too slowly. Increasing p_{max} TCP reacts appropriate to the loss situation and achieves the local maximum. Because of the further increasing p_{max} the number of successfully transmitted packets decreases.

5 Conclusion and Outlook

In this paper we present an analytical model of TCP data transmission over a RED queue. Early single packet loss of the RED queue gives feedback to the TCP sources to control their congestion window using slow start and congestion avoidance. Performance measures like distributions for queue occupancy and successfully transmitted packets are obtained. The distribution of the TCPs congestion window size and averaged RED queue occupancy is also derived.

The results of the study show that proper dimensioning of the RED queue parameters is crucial to achieve a benefit and not to experience drawbacks from the RED queue. The performance is optimized, when the variance of the TCPs congestion window is minimized.

Further studies will investigate the influence of non-linear loss functions for a RED queue, study the fairness of non homogeneous sources and extend the numerical results for a larger number of connections.

Acknowledgement

The authors would like to thank Dirk Staehle for the productive discussions of the presented results. The financial support of the T Nova, Deutsche Telekom AG (Technologiezentrum Darmstadt) is appreciated.

References

[1] W. Stevens, "RFC2001: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms." http://www.ietf.org/rfc/rfc2001.txt, Jan. 1997.

[2] B. Braden, D. Clark, J. Crowcroft, and et al, "RFC2309: Recommendations on queue management and congestion avoidance in the internet." http://www.ietf.org/rfc/rfc2309.txt, Apr. 1998.

[3] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, Aug. 1993.

[4] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "RFC2597: Assured forwarding PHB group." http://www.ietf.org/rfc/rfc2597.txt, Jan. 1997.

[5] D. Lin and R. Morris, "Dynamics of random early dectection," *ACM SIGCOMM Computer Communications Review*, vol. 27, pp. 127–136, Oct. 1997.

[6] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," in *IEEE INFOCOM'99*, (New York, USA), Mar. 1999.

[7] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," in *IEEE INFOCOM'99*, (New York, USA), Mar. 1999.

[8] M. May, J.-C. Bolot, A. Jean-Marie, and C. Diot, "Simple performance models of differentiated services schemes for the internet," in *IEEE INFOCOM'99*, (New York, USA), Apr. 1999.

[9] T. Bonald, M. May, and J.-C. Bolot, "Analytic evaluation of RED performance," in *IEEE INFOCOM'2000*, (Tel Aviv, Israel), Apr. 2000.

[10] S. Peeters and C. Blondia, "A discrete time analysis of random early detection with responsive best-effort traffic," COST-257 Technical Document 257TD(99)29, COST-257 MC meeting, Cyprus, Sept. 1999.

[11] M. Allman, V. Paxson, and W. Stevens, "RFC2581: TCP congestion control." http://www.ietf.org/rfc/rfc2581.txt, Apr. 1999.

[12] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of SIGCOMM'98*, 1998.

[13] M. Menth, "Carrying wireless traffic over IP using realtime transport protocol multiplexing," in *12th ITC Specialist Seminar*, (Lillehammer, Norway), March 2000.

[14] M. Menth and N. Gerlich, "A numerical framework for solving discrete finite markov models applied to the AAL-2 protocol," in *MMB '99, 10th GI/ITG Special Interest Conference*, (Trier), pp. 0163–0172, Sep. 1999.

[15] P. Tran-Gia, "Discrete-time analysis technique and application to usage parameter control modeling in ATM systems," in *8th Australian Teletraffic Research Seminar*, (Melbourne), Dec. 1993.

[16] N. Vicari and S. Köhler, "Measuring internet user traffic behavior dependent on access speed," Tech. Rep. 238, University of Würzburg, Institute of Computer Science, Oct. 1999.

[17] "UCB/LBNL/VINT Network Simulator - ns (version 2)." Source code and documentation available at http://www-mash.cs.berkeley.edu/ns/.