ITC19/ Performance Challenges for Efficient Next Generation Networks 677-686
LIANG X.J. and XIN Z.H.(Editors)
V.B. IVERSEN and KUO G.S.(Editors)
Beijing University of Posts and Telecommunications Press

RELATIVE FREQUENCY OF TCP RETRANSMISSION TIMEOUTS FOR BURST SEGMENT LOSSES

Thorsten Müller [a], Jörg Schüler [b]

[a] Technische Universität Dresden,Chair for Telecommunications, 01062 Dresden, Germany
now with Vierling Communications GmbH, Ebermannstadt, Germany
Thorsten-Mueller@gmx.de

[b] Technische Universität Dresden,Chair for Telecommunications, 01062 Dresden, Germany
now with Radioplan GmbH, Dresden, Germany
joerg.schueler@radioplan.com

Abstract: Detection of segment drops is essential for performance modelling of TCP. Most TCP-variants uses two types of drop detection: Fast Retransmission and Retransmission Timeout. In this paper we will introduce a mathematical expression to determine the type of drop detection out of the transmission circumstances like TCP-variant, window sizes, buffer sizes and loss pattern – a criterion which was not mentioned in great detail up to now. This paper will show if enough segments are available causing dupacks for the initiation of Fast Retransmission, it will be started; if not, a Retransmission Timeout will be initiated. This behaviour is independent of the network structure. The investigation of this criterion is complex for bursty drops. This paper will give mathematical expressions to find the series of drop detection with relation to the above mentioned factors. In addition the results were checked by a simulator TCPSim.
Keywords: Transmission Control Protocol - TCP, Performance Modelling, Protocol Analysis

1 Introduction

TCP [1] is a widely used protocol in todays Internet. Many scientists searched for models for the performance of TCP [2]. One important influence is the drop detection of TCP. The type of drop detection influence the following transmission rate. Most TCP-variants uses two types of drop detection: Fast Retransmission and Retransmission Timeout [3]. In this paper we focus on TCP-Reno, represented by RFC 2581 [4]. It is one of the most common variants of TCP and is the basis for several extensions like TCP-New-Reno, SACK, TCP-Westwood etc. This paper uses the relevant RFCs as the only information source. No (freely) available simulator or implementation is used for the analysis.
In section 2 we prove that only the present of enough duplicate acknowledgements determine the type of drop detection. In section 3 we define several variables to describe exactly our analysis. In section 4 and 5 the behaviour for one and two drops are analysed, respectively. In section 6 a generalized way of the analysis is presented. Section 7 shows the results – over all usually observed window – sizes for three and four drops. An automated way of analysis with an own written analyser is used. Section 8 finishes the paper.

2 Basic Relations

The type of drop detection (Fast Retransmission or Timeout) is an essential point for performance modelling of TCP. This decision can be reduced to a simple equation.

$$\text{number of dupacks} \geq \mathbf{X} \tag{1}$$

If the equation is NOT fulfilled a Timeout will occur, if the equation is fulfilled a Fast Retransmission will occur. In the first moment it is surprising that only one simple criterion is necessary for this decision. Therefore this criterion should be examined in greater detail.
The RTO-timer value tends to RTT-value for long-run connections with a steady Round-Trip-Time. This would lead to Retransmission Timeouts for packets when some latency is shortly added to the network structure. Preventing such a scenario RFC 2988 [5] gives the advise to set the variation of the measured RTT-values at a minimum which is equal to the granularity G[1] (one tick) of the clock used by TCP. A second advise is to set the RTO timer not lower than $(2.5 + \mathrm{G})$ seconds.
Another important fact, the RTO timer is set back every time a packet is sent into the network by the sender. There is normally only one RTO timer for each TCP connection, so the timer observes only the latest packet with the actual RTO-value, all other packets have more time.
This behaviour of the RTO-timer causes an absence of the Retransmission Timeout if packets were sent into the network. If there are enough packets in the network for a Fast Retransmission, it will occur; if there are not enough packets a Retransmission Timeout will occur. This explains this simple and clearly defined criterion for the detection of a drop.

3 Definitions

For TCP reaction analysis on bursty segment drops the following terms and definitions are applied:

transmission window $[W_m]$ *:* Size of the allowed TCP transmission window after reacting on the detection of segment drop m. The unindexed variable W refers to the transmission window size at the time right before the first segment drop has been detected and is denominated as *error window*. $W_m = \min(C_m, R_m),\ m = 1..N$

congestion window $[C_m]$ *:* Size of the congestion window after reacting on the detection of segment drop m. The unindexed variable C describes the window size before the first segment drop detection. Note, that the congestion window may additionally be modified during other acknowledgement arrivals.

receiver window $[R_{mx}]$ *:* Unoccupied portion of the segment buffer at the TCP receiver[2] as a result of segment drop m. The variable x indicates whether the window size refers to a point in time before the reception of an successful retransmission ($x = a$) or after it ($x = b$). The unindexed variable R refers to the maximum possible receiver window which is the total *receiver buffer*.

loss pattern $\left[{}^{W}_{N}L_k\right]$ *:* Positions of dropped segments within an error window. With N as the

[1] In many implementations the granularity G is equal to 500ms.

[2] This analysis assumes non-blocking applications. Segment buffering is exclusively performed in case of missing data.

pre-defined count of segment drops within the error window. By definition of the error window, the first error is always located at the front position. ${}^{W}_{N}L_k = \{l_1\text{–}l_2\text{–}...\text{–}l_N\}$

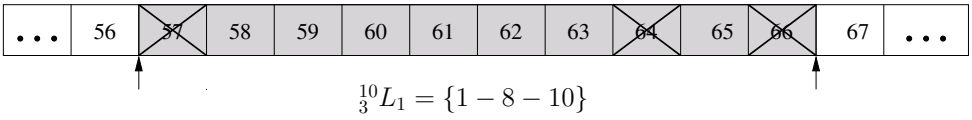

Figure 1: Example of a loss pattern

detection pattern $\left[B^N\right]$ *:* Sequence of TCP error detection schemes for the experienced segment drops. $B^N = \{B^N_1..B^N_N\}$ with $B^N_m = \{'\text{T}', '\text{F}'\}$ and $'\text{T}'-$ Retransmission Timeout, $'\text{F}'-$ Fast Retransmit and Recovery. Since TCP Retransmission Timeout performs a complete 'Go-Back-N' procedure refreshing the entire transmission state, sequencing packet drops are automatically retransmitted and not longer part of the investigated error window[3].

$$B^1 = \{\,'\text{T}',\ '\text{F}'\,\} \qquad B^3 = \{\,'\text{T}',\ '\text{FT}',\ '\text{FFT}',\ '\text{FFF}'\,\}$$
$$B^2 = \{\,'\text{T}',\ '\text{FT}',\ '\text{FF}'\,\} \qquad B^4 = \{\,'\text{T}',\ '\text{FT}',\ '\text{FFT}',\ '\text{FFFT}',\ '\text{FFFF}'\,\}$$

segment identifier $[o_m]$ *:* Absolute sequence number of the dropped segment m within the segment stream. $o_m = (o_1 - 1) + l_m$

duplicate acks $[D_m]$ *:* Final number of duplicate acknowledgements being received at the sender for segment drop m. If $D_m \geq X$, the TCP Fast Retransmit algorithm will be initiated. For the presented study addressing TCP Reno $X = 3$.

transmission deadlock : Situation, where the number of corresponding duplicate acknowledgements does not reach X to initiate a Fast Retransmit. Generally, a transmission deadlock can be caused by (1) an insufficient congestion window or (2) an insufficient receiver window in combination with an inauspicious number of in-flight segments.

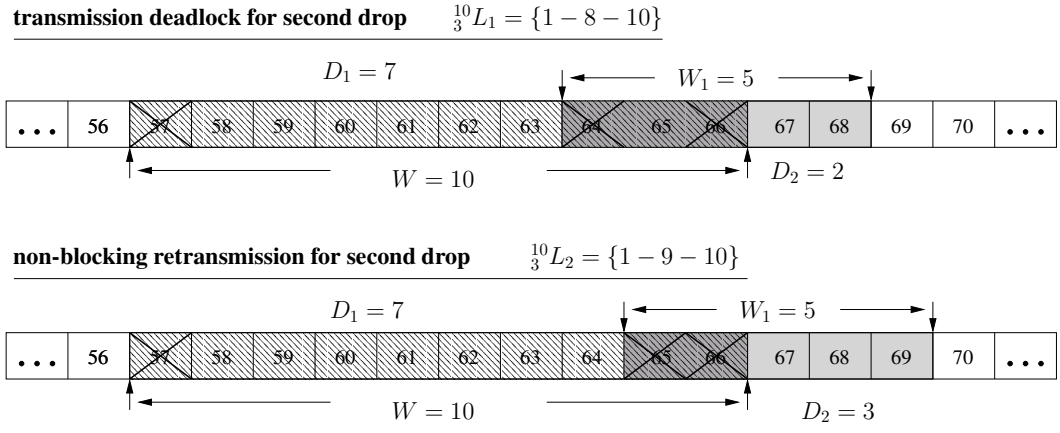

Figure 2: Transmission deadlock vs. non-blocking retransmission

[3]In that case, the detection pattern sequence is truncated

sender stopping segment $[S_m]$ *:* The sequence number of the earliest segment, which would cause the sender to run into a transmission deadlock (Retransmission Timeout). The value is determined for each particular segment drop m.

deterministic term $[T_{my}]$ *:* Auxiliary variable indicating the sequence number of a segment, which could possibly become the sender stopping segment. Different deterministic terms are addressed by $y = a, b, c...$, while m addresses the corresponding segment drop number.

Note, all window sizes, buffer sizes and sequence numbers of this investigation are applied in segment counts. This simplification has been introduced for the sake of comprehensibility and does not effect the validity of results.

4 TCP-Reno error detection of a single segment drop

In order to introduce the declared terms, this section discusses the analysis method for the very facile case of one segment drop ${}_1^W L_k$ occurring within an error window. Figure 3 illustrates this situation.

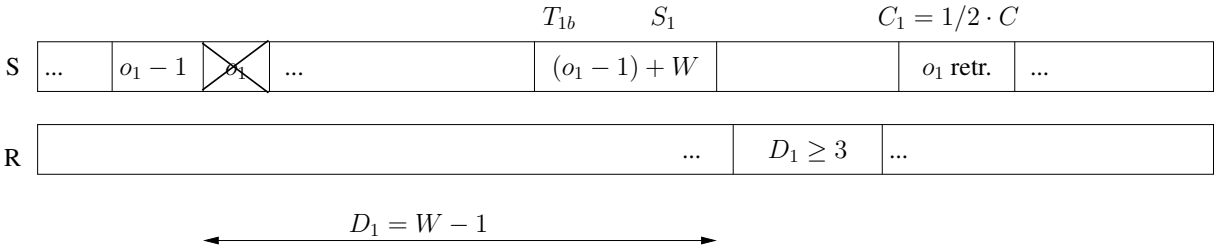

Figure 3: Condition analysis for single segment drop detection

As already mentioned, a Fast Retransmit $B^1 = \text{'F'}$ is always initiated for:

$$D_1(S_1) \geq X \tag{2}$$

Since the dropped segment is located at the front position of the error window[4], the number of sequencing segments arriving at the receiver and causing duplicate acknowledgements at the sender will be

$$D_1 = W - N \tag{3}$$

With W being either determined by the congestion window C or the receiver window R_{1a}, two deterministic terms T_{1a}, T_{1b} can be expressed[5]. Both point at the last segment of the potential error window:

$$T_{1a} = o_1 - 1 + R_{1a} \qquad T_{1b} = o_1 - 1 + C \tag{4}$$

[4]according to the definition of loss pattern

[5]Note, the calculation of deterministic terms and the sender stopping segment seems to be circumstantial for this evident example. Nevertheless, this method has been proven to be very helpful in complex window scenarios as discussed in the next section.

Applying $W = \min(C, R_{1a})$ to the deterministic terms (see RFC 2581 [4]), the sender stopping segment S_1 will be:

$$S_1 = \min(T_{1a}, T_{1b}) \tag{5}$$

According to our analysis model, the error window W is only defined for TCP states which have been fully recovered from previous disturbances. This leads to the fact, that the receiver buffer cannot contain any lacks of outstanding segments and must therefore be completely empty[6] ($R_{1a} = R$). Considering the receiver buffer size R not to be the limiting parameter for the initial error window[7], S_1 can be reduced to:

$$S_1 = T_{1b} = o_1 - 1 + W \qquad \text{with} \qquad W = C \tag{6}$$

The number of duplicate acknowledgements can now be expressed in absolute positions within the segment flow:

$$D_1(S_1) = S_1 - (o_1 - 1) - N \tag{7}$$

Applying equation (6) this leads for $N = 1$ to the consistent condition for a non-blocking detection of a single segment drop:

$$B^1({}^{W}_{1}L_k) = \begin{cases} \text{'T'} : & W < X + 1 \\ \text{'F'} : & W \geq X + 1 \end{cases} \qquad \forall k \tag{8}$$

5 TCP-Reno error detection of two segment drops

The occurrence of two drops ${}^{W}_{2}L_k$ within a single error window has been investigated case-sensitive for each missing segment. For the first segment drop l_1 the considerations of the previous section can directly be applied. With a total number of errors $N = 2$, the TCP error detection will be performed according to:

$$B^2_1({}^{W}_{2}L_k) = \begin{cases} \text{'T'} : & W < X + 2 \\ \text{'F'} : & W \geq X + 2 \end{cases} \qquad \forall k \tag{9}$$

For the second segment drop l_2 the situation becomes more complex. It can either directly be detected by duplicate acknowledgements from segments placed in the initial error window. Or it might be the case, that additional segments raised after retransmission of the first segment drop l_1 trigger the required duplicate acknowledgements. Both, window adaptation schemes and receiver buffer control mechanisms come now into consideration. Figure 4 visualizes the analytical considerations.

In order to calculate the sender stopping segment S_2, the deterministic terms T_{2y} have to be calculated twice; before (T_{2a}, T_{2b}) and after (T_{2c}, T_{2d}) the detection of l_1. Considering the

[6] See definition of receiver window

[7] This consideration was applied in context of other investigations. It must not necessarily be true for all scenarios.

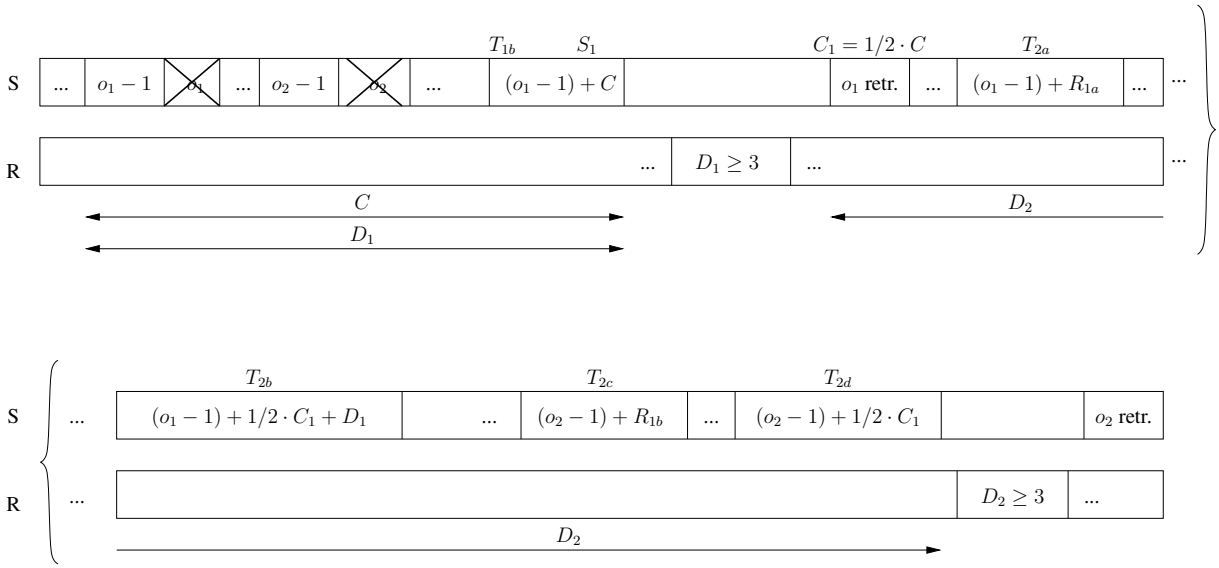

Figure 4: Condition analysis for two segment losses detection

maximum of both potential scenarios now:

$$S_2 = \max\left[\min(T_{2a}, T_{2b}), \min(T_{2c}, T_{2d})\right] \tag{10}$$

with

$$\begin{array}{lcllcl} T_{2a} & = & (o_1 - 1) + R_{1a} & T_{2b} & = & (o_1 - 1) + C_1 + D_1 \\ T_{2c} & = & (o_2 - 1) + R_{1b} & T_{2d} & = & (o_2 - 1) + C_1 \end{array} \tag{11}$$

Term T_{2b} is increased by the number of received duplicate acknowledgements for the first segment drop. This is caused by the inflating window mechanism of TCP Reno [4]. In the same acknowledgement the free space at the receiver buffer is transmitted to the sender. Each segment, sent after the drop o_1, arrived at the receiver releases one dupack and shrinks the free receiver buffer R_{1x} by one segment.

$$\begin{array}{lcllcl} C_1 & = & \lfloor 1/2 \cdot W \rfloor & D_1 & = & W - N \\ R_{1a} & = & R - D_a & R_{1b} & = & \lfloor 1/2 \cdot R + 1/2 \cdot N \rfloor \end{array} \tag{12}$$

D_a expresses the increasing number of dupacks **and** the used space at the receiver buffer. The sender will stop sending segments, if the receiver buffer has not enough space. This will be the case for

$$D_a = 1/2 \cdot R - 1/4 \cdot W + 1/2 \cdot N \tag{13}$$

and T_{2a} becomes smaller than T_{2b}. For each drop space is reserved at the receiver buffer – causing the term $1/2 \cdot N$.

This leads to the deterministic terms[8]:

$$\begin{array}{lcl} T_{2a} & = & (o_1 - 1) + \lfloor 1/2 \cdot R + 1/4 \cdot W - 1/2 \cdot N \rfloor \\ T_{2b} & = & (o_1 - 1) + \lfloor 3/2 \cdot W \rfloor - N \\ T_{2c} & = & o_2 + \lfloor 1/2 \cdot R + 1/2 \cdot N \rfloor \\ T_{2d} & = & o_2 - 1 + \lfloor 1/2 \cdot W \rfloor \end{array} \tag{14}$$

The number of duplicate acknowledgements generated for the second segment drop will be:

$$D_2(S_1, S_2) = S_2 - S_1 \qquad \text{with} \qquad S_1 = T_{1b} = o_1 - 1 + W \tag{15}$$

This set of equations can now be applied to the general condition for TCP error detection leading to:

$$B_2^2({}_2^W L_k) = \begin{cases} \text{'T'} : & D_2(S_1, S_2) < X \\ \text{'F'} : & D_2(S_1, S_2) \geq X \end{cases} \qquad \forall k \tag{16}$$

Assuming an exemplary parameter set of $R = 37$, $W = 20$ and $X = 3$, the above equations give

$$D_2(S_1, S_2) = \max[1, (o_2 - o_1 - 10)] \tag{17}$$

and finally the following detection pattern:

$$B^2({}_2^{20} L_k) = \begin{cases} \text{'T'} : & \emptyset \\ \text{'FT'} : & {}_2^{20} L_k \in \{1,2\}, \{1,3\} ... \{1,13\} \\ \text{'FF'} : & {}_2^{20} L_k \in \{1,14\}, \{1,15\} ... \{1,20\} \end{cases} \tag{18}$$

6 General procedure

The previously introduced method used for the analysis of TCP detection pattern on single and double segment drops is applicable without restrictions for arbitrary loss patterns ${}_N^W L$ ($N = 1..W, W{=}1..\infty$). Nevertheless, the description of the deterministic terms T_{my} is subtle and can not be achieved by straight forward rule assimilation. The complex interactions between window adaptation rules, buffer management and network queueing require a precise investigation of each potential scenario.

Even though only one of the deterministic terms finally describes the sender stopping segment for a particular drop m, a general approach for complexity reduction could not have been found. Each combination of the terms has to be evaluated to find a closed description for $D_m({}_N^W L, S_1..S_m)$ with $S_m(W, R)$. The generalized way of finding the detection pattern is shown in figure 5. The complexity of the problem with $\sigma = 4^{N-1}$ is a direct function of the number of segment drops N, caused by the four terms for each drop, except the first.

[8]Note, that the necessary condition $W \leq R$ leads in this case to the reduction of term T_{2c}.

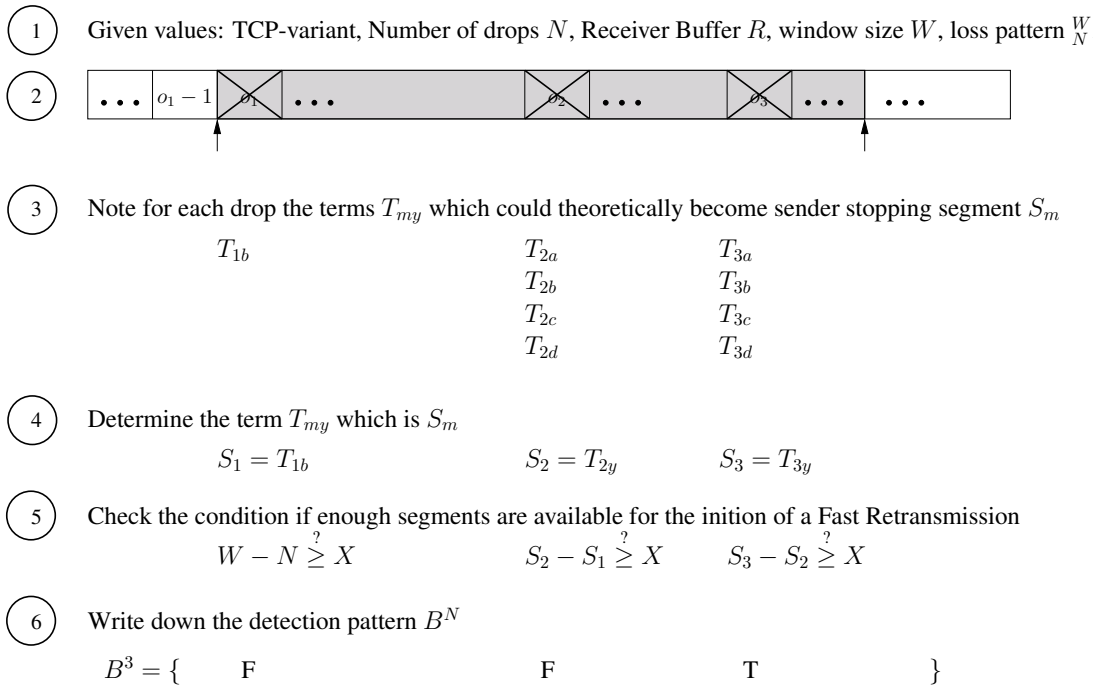

Figure 5: General procedure analysing the way of error detection

7 Results

In the previous sections mathematical expressions have been presented for the evaluation of error detection pattern B^N in reaction on segment losses ${}^{W}_{N}L$. In order to validate the analytical results, simulation studies of the TCP behaviour have been performed. Also, the exploding number of deterministic terms for segment drop counts $N > 3$ restricts so far the practicability of the proposed analytical method in such cases.

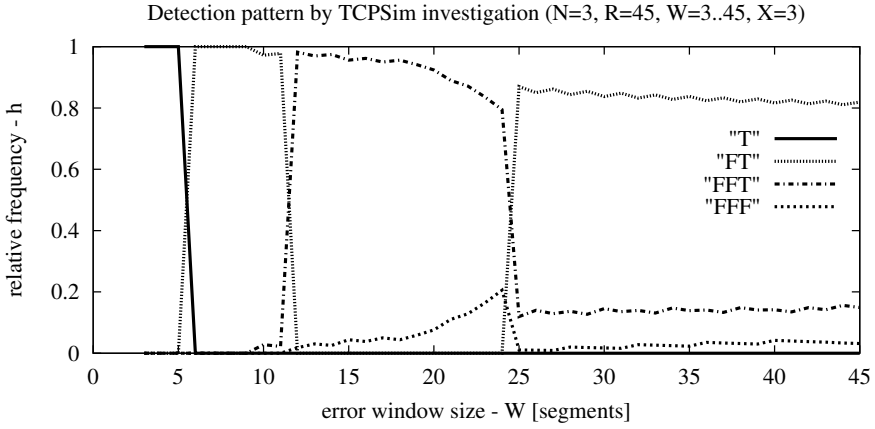

Figure 6: Relative frequency of detection pattern B^3 for TCP Reno (TCPSim)

In favour of a maximum conformance to all existing RFC specifications, the simulation studies have been performed with the proprietary tool *TCPSim*[9] [6]. The decision not to use the

[9] *TCPSim* was developed at Technische Universität Dresden, Chair for Telecommunications, Germany.

well-known *ns-2* simulator [7] was raised by the fact, that it does not model the influence of the receiver window size appropriately. Since the analytical investigations have indicated a strong impact of this parameter, comparability to the simulation results would have been questionable using *ns-2*. With *TCPSim* the complete equivalence of results has been proven for N=2, W=2..45, R=40, 45) and indicated for many cases with N=3.

Figure 6 gives the frequency of detection pattern occurrence $h(B^3)$ over initial error window size W. The frequency is normalized by the number of loss pattern permutations k. In figure 7 the corresponding results for a number of segment drops $N = 4$ is shown. Both studies have been performed with a receiver buffer size $R = 45$.

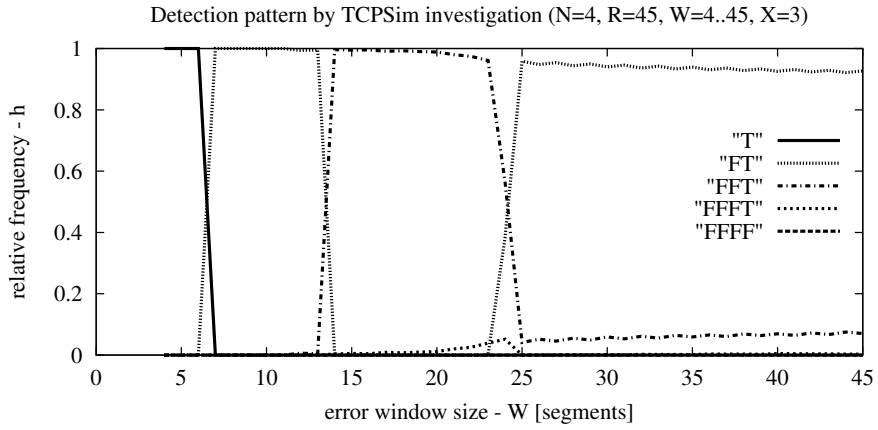

Figure 7: Relative frequency of detection pattern B^4 for TCP Reno (TCPSim)

Both figures emphasize the expected trend, that with increasing window size more segment losses can be detected by Fast Retransmit. This is caused by a higher number of potential duplicate acknowledgements. Below $W = X + N$ the required amount $D_1 = X$ can not be reached and already the first drop causes a Retransmission Timeout in any case. Error window sizes between $X+N \leq W < \lfloor 2 \cdot (X+N) \rfloor$ can perform Fast Retransmit detection for the first segment but suffer at the second. Nevertheless, there exist a few drop combinations ${}^{W}_{N}L$, which already can overcome the Timeout for the second drop. Further increase of W leads to a guaranteed double Fast Retransmit. Also, the possibility for a triple Fast Retransmit detection increases.

For $W \geq \lfloor R/2 \rfloor + 2$ the influence of the receiver window becomes predominant. Surprisingly, the number of segment drops being detected with Fast Retransmit jumps back to two for the majority of all error patterns. The reason is, that a receiver window $R_{3a} \leq C_2$ is acknowledged, causing the sender not to release any more data, since C_2 segments are already in-flight. Because all duplicate acknowledgements for l_2 refer to the last in-order segment and not to its own sequence number, the sender is not able to estimate the number of in-flight segments already stored in the receiver buffer. For $C_2 \geq \lfloor R/2 \rfloor$ the funny situation occurs, that the receiver buffer has $R - C_2$ free slots available, but the sender cannot release any more data since it must assume that C_2 segments are still on the way. The indication of this self-blocking nature of the TCP protocol is one of the major outcomes of this work.

8 Conclusion

We show mathematical expressions and a more automated way (TCPSim) to examine the reaction of TCP-Reno to bursty drops. We detected two facts – loss pattern and free receiver buffer – which should receive more attention in following experiments and examinations of TCP.
We observed another interesting effect during our work with TCPSim: When you have a sequence of three drops followed by three good transmitted packets and the congestion window is set to one before the sequence of these six packets is released, you can observe that the sender will detect drops for packets which are absolutely correctly transmitted. The reason is the receiver buffer which stores the good transmitted packets when a drop occur. These stored packets leads to several dupacks and releases three Fast Retransmissions like mentioned above. The packets detected as drops has nothing to do with the six packets transmitted before.

References

[1] R. Braden, "RFC 1122: Requirements for internet hosts - communication layers." `http://www.ietf.org/rfc/rfc1122.txt`, 1989.

[2] C. Casetti and M. Meo, "Modeling the stationary behavior of tcp reno connections," in *Proceedings of the International Workshop on Quality of Service in Multiservice IP Networks*, pp. 141–156, Springer-Verlag, 2001.

[3] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1998.

[4] M. Allman, V. Paxson, and W. Stevens, "RFC 2581: TCP congestion control." `http://www.ietf.org/rfc/rfc2581.txt`, April 1999.

[5] V. Paxson and M. Allman, "RFC 2988: Computing TCP's retransmission timer." `http://www.ietf.org/rfc/rfc2988.txt`, 2000.

[6] J. Schueler and T. Mueller, "TCPSim." available from May 2005 at `http://www.ifn.et.tu-dresden.de/TK`, December 2004.

[7] "The network simulator - ns-2." Web page: `http://www.isi.edu/nsnam/ns/`, July 2002.