The Sensitivity of TCP to Sudden Delay Variations in Mobile Networks

Michael Scharf, Marc Necker, and Bernd Gloss

Institute of Communication Networks and Computer Engineering
University of Stuttgart, Germany*
{scharf,necker,gloss}@ikr.uni-stuttgart.de

Abstract. This paper studies the impact of variable transmission delays on the Transmission Control Protocol (TCP). Sudden delay variations, which are not uncommon in mobile networks, may degrade the performance since they may cause spurious TCP timeouts. The most important parameter in this context is the TCP retransmission timer. In this paper, we analyze TCP's round-trip time estimation for bulk data traffic over wireless links. The main contribution is a new analytical model that accurately predicts the timeout duration from given network parameters. As a first result, the model shows that the round-trip time sampling rate has a significant impact on the timer characteristics. Therefore, the standardized estimation algorithm does not harmonize well with timestamp-based measurement. Second, we quantify the risk of spurious TCP timeouts triggered by changing round-trip times, in particular long off periods. We conclude that delay variations are only critical when they are on the order of seconds.

1 Introduction

The Transmission Control Protocol (TCP) [1] is widely used in the Internet as a reliable end-to-end transport protocol. Mobile Internet access will be one of the key features in networks like the General Packet Radio Service (GPRS) or the Universal Mobile Telecommunication System (UMTS). Their link layer shields the transport layer from data corruption in the radio channel by using forward error correction (FEC) and automatic repeat request (ARQ) mechanisms. However, this error protection comes at the cost of higher latencies and jitter. Further effects in cellular networks are handovers, link outages and radio resource preemption caused by high-priority voice traffic. Thus, the data transmission may suffer from sudden delay variations or so-called "delay spikes" [2, 3].

TCP relies on the retransmission timeout (RTO) to detect packet loss. The sender measures the round-trip time (RTT) and dynamically adapts the RTO value as specified in RFC 2988 [4], taking into account both the low-pass filtered RTT samples and the observed delay variance. Sudden delay variations

* The research work of this paper was done in cooperation with Alcatel SEL AG, Research and Innovation Department, Lorenzstr. 10, 70435 Stuttgart, Germany.

are problematic since they may trigger a spurious timeout even if no packets are lost. This affects TCP performance in two ways: First, the TCP sender retransmits outstanding packets due to the *Go-back-N* mechanism; and second, the congestion window and thereby the sending rate are unnecessarily reduced. As a consequence, TCP may waste scarce bandwidth or underutilize the available resources. Both effects have already been studied extensively [5–10].

The most important parameter in this context is the TCP retransmission timeout. However, due to the complexity and the cumulative nature of the RTT estimation, this particular aspect of TCP has rarely been addressed so far. In this paper we give a detailed insight into the characteristics of the RTT estimation, both with and without timestamp-based measurement. We determine upper and lower bounds for the RTO duration as a function of the network configuration. This allows us to quantify the sensitivity of TCP to sudden delay variations. Unlike previous studies [11, 12], we use an analytical approach and focus on networks with rather low data rates and high latencies. To the best of our knowledge, this is the first analytical model of the RTT estimator in TCP.

The remainder of the paper is organized as follows. In Section 2, we develop a delay model based on the saw-tooth behavior of bulk data traffic and describe the round-trip time measurement in TCP. Section 3 investigates the performance of the RTO estimation for this delay model using digital signal processing theory. In Section 4, we verify our analysis by simulation and discuss the impact of several parameters. Based on our model, Section 5 evaluates the risk of spurious timeouts. Finally, Section 6 concludes the paper and summarizes our results.

2 An End-to-End Delay Model

2.1 TCP in Networks with a High Bandwidth-Delay Product

Since significant delay variations usually come along with transient events like handovers, we first focus on the rather stationary situation between such events in order to determine the expected RTO duration. An analytical analysis of bulk data TCP traffic over a single bottleneck link has been presented in [13]. The underlying model is illustrated in Fig. 1. Some assumptions are: (1) There is one dedicated drop-tail buffer of fixed size B in front of the bottleneck link, (2) it is not necessary to consider two-way traffic, (3) all data packets are of equal size L_{MTU} ($= 1500\,\mathrm{byte}$), and (4) all delays except for the service time and the queuing delay at the bottleneck link can be lumped together to single latency τ. When accessing the Internet through a cellular network, the radio link is likely to be the bottleneck. We thus argue that this model reflects the characteristics of 2.5 G and 3 G networks, in particular in downlink direction, as long as both the service rate μ and the latency τ remain at an approximately constant level.

In the network model depicted in Fig. 1, the congestion window W_{C} of a single TCP connection with greedy source follows a saw-tooth pattern because of the congestion control [14]. The sender goes into the *Congestion Avoidance* after having probed the available bandwidth during an initial *Slow Start*. In

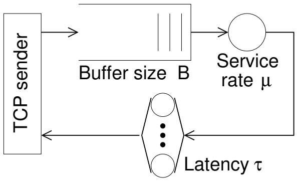

Fig. 1. End-to-end path model

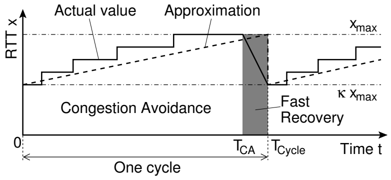

Fig. 2. Typical RTT evolution

Congestion Avoidance, W_C is gradually increased until the capacity of the path

$$C = \lfloor B + 1 + \mu\,\tau \rfloor \tag{1}$$

is exceeded and a single packet is dropped. Both B and W_C are here counted in segments. When detecting this packet loss, the sender enters the *Fast Recovery*, retransmits the missing segment, and approximately halves W_C from $W_\mathrm{max} = C + 1$ to $\frac{1}{2}\,W_\mathrm{max}$. On receiving new acknowledgments (ACKs), W_C is increased again. The congestion window thus follows a regular pattern that can be divided into "cycles" [13]. Note that the saw-tooth behavior may be prevented by a small receiver-advertised window W_R. However, W_R is unlikely to be a limiting factor if the *window scale option* [15] is used, as recommended in [16].

The evolution of W_C in one cycle is determined in [13]. We extend this analysis in two aspects. First, the assumption that B is smaller than the minimal bandwidth-delay product $\mu\,\tau + 1$ contradicts current buffer overprovisioning in mobile networks [2, 3]. As a consequence, we rather assume $B > \mu\,\tau + 1$. Second, we do not neglect the *delayed acknowledgments* in TCP [14]: A TCP receiver should only generate an ACK for every second full-sized segment. Further on, ACKs must not be delayed for more than a certain duration d, which many TCP implementations set to 200 ms [17]. Thus, if $\mu < 1/d$, there is one ACK per received segment ($b = 1$). This corresponds to a data rate $r = \mu\,L_\mathrm{MTU} < 60$ kbps. The delay d affects the total latency τ by an additional component $\tau_\mathrm{DelAck} = d$. For higher data rates, only every second segment is acknowledged, i. e., $b = 2$.

The raise of W_C from $\frac{1}{2}\,W_\mathrm{max}$ to W_max during *Congestion Avoidance* can be described by the differential equation $\frac{\mathrm{d}W_\mathrm{C}}{\mathrm{d}t} = \frac{\mathrm{d}W_\mathrm{C}}{\mathrm{d}a} \cdot \frac{\mathrm{d}a}{\mathrm{d}t}$, where $\frac{\mathrm{d}W_\mathrm{C}}{\mathrm{d}a} \approx \frac{1}{W_\mathrm{C}}$ denotes the rate of window growth per ACK, and $\frac{\mathrm{d}a}{\mathrm{d}t} \approx \frac{\mu}{b}$ is the ACK arrival rate (see [13]). W_C in the first phase of one cycle ($0 \leq t \leq T_\mathrm{CA}$) is thus given by

$$W_\mathrm{C}(t) = \sqrt{\frac{1}{4}\,(C+1)^2 + \frac{2\,\mu\,t}{b}}\ . \tag{2}$$

2.2 Modeling the Round-Trip Time Measurement

While W_C raises, the buffer gradually fills up. At the same time the queuing delay increases and so does the round-trip time x. More precisely, the RTT is always incremented by $1/\mu$ when the congestion window advances by a full-sized segment. Provided that jitter can be neglected, x is a step-wise increasing function [18]. As sketched in Fig. 2, it roughly follows the square-root law given by (2). Apparently, the maximum value is defined by $x_{\max} = C/\mu$. When entering the *Fast Recovery* at the end of the cycle, W_{C} is halved and the buffer drains. This reduces the RTT by a factor of two.

The evolution of the round-trip time as shown in Fig. 2 is not a very useful function. In the following, we ignore it and instead use a linear approximation: If there are N samples in one cycle, we assume these samples $x(n)$ as

$$x(n) = \kappa\, x_{\max} + \hat{\kappa}\, x_{\max} \frac{n}{N-1} \tag{3}$$

for $0 \leq n \leq N-1$. As already discussed, the ratio κ $(= 1 - \hat{\kappa})$ between the smallest and the largest RTT can be set to $\frac{1}{2}$.

The missing parameter in (3) is the number of RTT samples N. TCP may use two different measurement methods: By default, the TCP sender records the time when a segment has been sent. On arrival of an ACK comprising this segment, the RTT is obtained from the difference of the system time to this stored variable. This means that there is always only one ongoing measurement. A more frequent measurement is possible if the timestamp option [15] is included in the TCP header. The sender can then derive a sample from every new ACK.

From an abstract point of view, both methods differ in the sampling rate. In the first case, one measurement is taken per RTT. This corresponds to

$$N_{\mathrm{default}} \approx b \left\lceil \frac{1}{2}\,(C+1) + 2 \right\rceil \tag{4}$$

samples in one cycle. In the case $b = 1$, this formula is straightforward: $\frac{1}{2}(C+1)$ is the number of increments in W_{C} occurring about once per RTT, just like the RTT measurement. The total number of samples is increased by two in order to account for the *Fast Recovery* and the transition back to *Congestion Avoidance*. For $b = 2$, only every second segment triggers an ACK, and $W(t)$ increases at a slower rate. This approximately doubles the number of samples.

With timestamps, every segment is timed during *Congestion Avoidance*. By solving (2) for $W_{\mathrm{C}}(T_{\mathrm{CA}}) = W_{\max}$, one obtains $T_{\mathrm{CA}} = \frac{3\,b}{8\,\mu}\,(C+1)^2$. During this time, $\mu\, T_{\mathrm{CA}}$ segments are transmitted. Taking further into account the retransmitted segment and one additional window of data, which is sent before the packet loss is detected, the number of RTT samples can be approximated by

$$N_{\mathrm{timestamps}} \approx \left\lceil \frac{3}{8}\,(C+1)^2 + \frac{C}{b} + 1 \right\rceil \quad . \tag{5}$$

Note that for $b = 2$ the number of measurements is approximately halved because there is only one ACK for every other segment.

3 Performance of the Round-Trip Time Estimator

3.1 Mathematical Analysis

Setting the retransmission timeout is a challenging issue. On the one hand, an aggressive algorithm tends to trigger frequent spurious timeouts. On the other hand, a conservative timer may cause long idle times in case of packet loss. TCP tries to achieve a trade-off between both extremes by calculating the RTO duration $R(n)$ for every new sample $x(n)$ out of exponentially weighted moving averages for the smoothed round-trip time $s(n)$ and the RTT variation $v(n)$:

$$\begin{aligned} v(n) =& \hat{\beta}\, v(n-1) + \beta \cdot \big|s(n-1) - x(n)\big| && (6)\\ s(n) =& \hat{\alpha}\, s(n-1) + \alpha\, x(n) && (7)\\ R(n) =& \max\big(s(n) + \gamma\, v(n), m\big) \quad . && (8) \end{aligned}$$

According to RFC 2988, the weighting factors should be set to $\alpha = 1 - \hat{\alpha} = \frac{1}{8}$, $\beta = 1 - \hat{\beta} = \frac{1}{4}$ and $\gamma = 4$. Furthermore, the lower bound m should be one second.

Given that the typical round-trip times in fixed networks are significantly lower than one second, $R(n)$ usually is dominated by m, while it is virtually unaffected by the other parameters [11]. In cellular networks, however, the RTT is on the order of several hundred milliseconds or even larger [2, 3]. This makes it important to understand the performance of Equations (6)–(8) in detail, in particular if the clock granularity of the operating system is small.

In principle, the computation of the RTO value from RTT samples is a digital signal processing problem, where $s(n)$ is a linear low-pass filter. By applying (3) as input function to (7), the smoothed RTT value can be calculated as

$$s(n) = x(n) + x_{\max}\, \big(c_1\, \hat{\alpha}^{n+1} - c_2\big) \qquad (9)$$

for $0 \le n \le N-1$, e.g. using the well-known Z-transform. Note that $s(n)$ differs from $x(n)$ by a diminishing component, which results from the step at the beginning of the cycle, and by a constant offset. Utilizing the periodicity $s(n) \equiv s(n+N)$ we obtain:

$$\begin{aligned} c_1 =& c_4 - \kappa + \frac{\hat{\kappa}}{\alpha\,(N-1)} && (10)\\ c_2 =& \frac{\hat{\kappa}\,\hat{\alpha}}{\alpha\,(N-1)} = c_3\,\hat{\alpha} && (11)\\ c_4 =& \frac{1}{1-\hat{\alpha}^N}\,\big(1 - c_2 - (\kappa - c_3)\,\hat{\alpha}^N\big) \quad . && (12) \end{aligned}$$

The smoothed RTT variance $v(n)$ is a non-linear function because of the absolute value in Equation (6). Nevertheless, $v(n)$ can be analyzed similarly with the help of $y(n) = \big|s(n-1) - x(n)\big|$ being piecewise defined as follows:

$$\frac{y(n)}{x_{\max}} \approx \begin{cases} c_1\,\hat{\alpha}^n - c_3, & \text{if} \quad 0 \le n < k \ , \\ c_3 - c_1\,\hat{\alpha}^n, & \text{if} \quad k \le n \le N-1 \ . \end{cases} \qquad (13)$$

Therein, $k = \ln \frac{c_3}{c_1} / \ln \hat{\alpha}$ is given by the solution of $y(k) = 0$. Applying (13) to $v(n) = \hat{\beta}\, v(n-1) + \beta\, y(n)$ yields the result

$$\frac{v(n)}{x_{\max}} = \begin{cases} -c_3 + (c_5 + c_3)\hat{\beta}^{n+1} + c_1\,\beta\, \frac{\hat{\alpha}^{n+1} - \hat{\beta}^{n+1}}{\hat{\alpha} - \hat{\beta}}, & \text{if} \quad n < k \ , \\ c_3 + \big(v(l) - c_3\big)\hat{\beta}^{n-l} - c_1\,\hat{\alpha}^k\,\beta\, \frac{\hat{\alpha}^{n-l} - \hat{\beta}^{n-l}}{\hat{\alpha} - \hat{\beta}}, & \text{if} \quad n \ge k \ , \end{cases} \qquad (14)$$

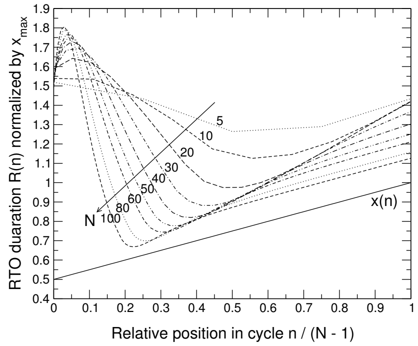

Fig. 3. Timeout duration in one cycle

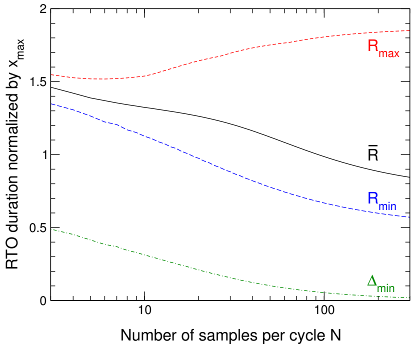

Fig. 4. Characteristic RTO values

where $l = k - 1$ is used as abbreviation. Again, c_5 can be calculated by taking into account $v(n) \equiv v(n+N)$:

$$c_5 = \frac{1}{1-\hat{\beta}^N} \left(c_3 \left(1 - 2\,\hat{\beta}^{N-k} + \hat{\beta}^N \right) + c_1 \frac{\beta}{\hat{\alpha} - \hat{\beta}} \left(2\,\hat{\alpha}^k\,\hat{\beta}^{N-k} - \hat{\alpha}^N - \hat{\beta}^N \right) \right) \quad . \tag{15}$$

Finally, we can compute $R(n)$ by inserting (9) and (14) into (8).

3.2 Numerical Results

In Fig. 3, the function $R(n)$ is plotted for different values of N, under the assumption that it is not affected by the lower bound m. The shape of $R(n)$ depends on the sampling rate. If the number of measurements per cycle is small, $R(n)$ remains at an almost constant level and is always greater than the maximum RTT $x_{\max}$. For higher values of N, $R(n)$ steeply increases at the beginning of the cycle. This peak is due to the fact that the drop in the RTT from $x_{\max}$ to $\kappa\,x_{\max}$ turns into a sudden increase of the RTT variation $v(n)$. Subsequently, $R(n)$ decays to a value close to $x(n)$. Here, the exponential moving average does not "remember" information about the previous cycle and the maximum value $x_{\max}$. As a consequence, $s(n)$ converges to a value close to $x(n)$, and $v(n)$ converges to zero. The larger the sampling rate, the more noticeable is this effect.

In order to get a simpler view on the set of curves in Fig. 3, we define the following metrics in addition to the mean value $\overline{R}$:

$$R_{\max} = \max_{0 \le n < N} \bigl(R(n) \bigr) \tag{16}$$

$$R_{\min} = \min_{0 \le n < N} \bigl(R(n) \bigr) \tag{17}$$

$$\Delta_{\min} = \min_{0 \le n < N} \bigl(R(n) - x(n) \bigr) \quad . \tag{18}$$

These metrics are depicted in Fig. 4 as a function of N. Because of the peak, the maximum $R_{\max}$ increases with N. Contrary to this, both the mean $\overline{R}$ and the

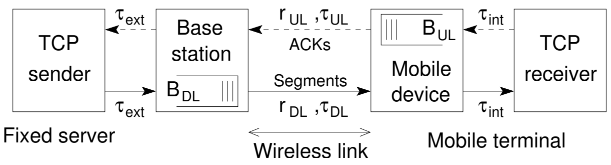

Fig. 5. Simulation topology

minimum $R_{\min}$ decrease. This also holds for $\varDelta_{\min}$ that gets very close to zero for $N \gg 10$. This means that the TCP sender is more aggressive when RTT measurement is based on timestamps.

The problem lies in the estimator weights α, β, and γ. They have been defined for the default RTT measurement taking at most one sample per RTT. Choosing adequate weights for the timestamps is an open research issue. One proposed solution is the *Eifel Retransmission Timer* [18] that adapts the parameters to the sampling rate. Also, newer versions of the Linux operating system deploy an algorithm that differs from the TCP specification [17]. Even though these empirical modifications address the issue of timestamp-based RTT measurement, a detailed analysis of their characteristics is still pending.

4 Model Validation

4.1 Simulation Setup

In order to compare our analysis with simulation results, we have modeled unidirectional TCP bulk data traffic over a mobile network using an object-oriented simulation tool [19]. Figure 5 shows the simulation setup.

The important simulation parameters are summarized in Table 1. We use several configurations both concerning the RTT estimator and the network model. The data rates in uplink and downlink direction are chosen from a set of values that are typical for cellular networks. As the link layer is supposed to be highly persistent, packet loss because of data corruption is not taken into account. In this section the radio channel is assumed to be rather ideal, i. e., it does not introduce a significant amount of jitter. B_{DL} is set to an integer value out of a range from 2 to 16 packets, which is equal to 3000–24,000 byte. In uplink direction we

Table 1. Simulation parameters

TCP parameter	**Value**
Algorithm	NewReno
L_{MTU} [byte]	1500
Delayed ACK d [ms]	200
Timestamps	disabled; enabled
Granularity G [ms]	10; 200
Min. RTO m [ms]	200; 1000
W_{R} [byte]	65536

Netw. param.	**Value**
(r_{DL}, r_{UL}) [kbps]	(13.4, 13.4); (53.6, 13.4); (64, 64); (128, 64); (384, 64)
τ_{ext} [ms]	10; 100
$\tau_{\mathrm{DL}} = \tau_{\mathrm{UL}}$ [ms]	100
τ_{int} [ms]	10
B_{DL} [pkts]	2–16
B_{UL} [pkts]	unlimited

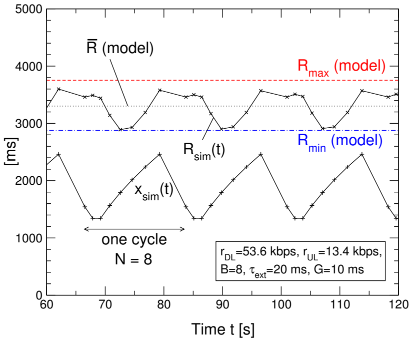

Fig. 6. Default RTT measurement

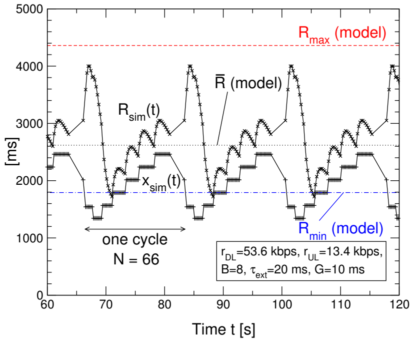

Fig. 7. Timestamp-based sampling

neglect the impact of finite buffer space because of the small size of acknowledgments $L_{\mathrm{ACK}} = 40\,\mathrm{byte}$ (52 byte with timestamp option). The corresponding parameters of the analytical model can easily be derived: Obviously, the service rate is $\mu = L_{\mathrm{MTU}}/r_{\mathrm{DL}}$, and B is equal to B_{DL}. Summing up all latencies yields $\tau = L_{\mathrm{ACK}}/r_{\mathrm{UL}} + \tau_{\mathrm{DL}} + \tau_{\mathrm{UL}} + 2\,(\tau_{\mathrm{ext}} + \tau_{\mathrm{int}}) + \tau_{\mathrm{DelAck}}$.

4.2 Simulation Results and Discussion

In the following, we first discuss the accuracy of our analytical model for two selected sample configurations. Figure 6 compares the RTO trace obtained from a simulation with the results of the analytical model. x_{sim} has approximately the expected saw-tooth characteristic with $N = 8$ samples per cycle, which corresponds to N_{default} as given by (4). R_{sim} remains within the bounds $R_{\max}$ and $R_{\min}$ determined from the analytical model, even though our model slightly overestimates the maximum. This value depends very much on the shape of the RTT reduction at the beginning of the cycle. From Fig. 6 it follows that, instead of immediately dropping from $x_{\max}$ to $\kappa\, x_{\max}$, as assumed in our analysis, there may be a sample in between. A detailed investigation reveals that this RTT sample is taken immediately after having completed the *Fast Recovery* and measures the RTT of segments released during the *Fast Recovery*.

A further example in Fig. 7 shows the same scenario with RTT measurement based on timestamps. Obviously, the number of RTT samples increases and changes the characteristics of the RTT estimator. Again, $N_{\mathrm{timestamps}} = 66$ from (5) perfectly matches the simulation results, and so does the minimum $R_{\min}$. Regarding the maximum, there is a similar effect as in the previous example. However, the most important difference is that the measured RTT x_{sim} only jumps up if W advances by one. Every increment affects the RTT variation $v(n)$ and thus results in a small peak. Since we do not consider this phenomenon in our linear approximation, we may underestimate the mean round-trip time.

In order to validate our model for a larger parameter space, we performed the same analysis for all parameter combinations of r_{DL}, r_{UL}, B_{DL}, and τ_{ext}, as

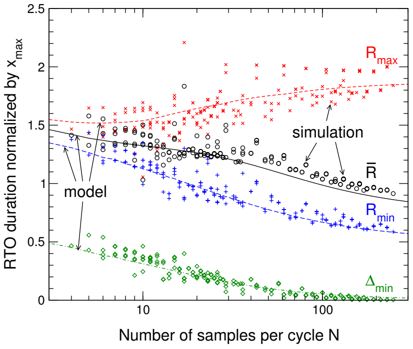

Fig. 8. Comparison model/simulation for $G = 10\,\text{ms}$ and $m = 200\,\text{ms}$

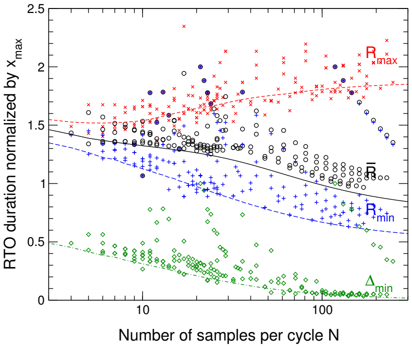

Fig. 9. Comparison model/simulation for $G = 200\,\text{ms}$ and $m = 1\,\text{s}$

listed in Table 1. $R_{\max}$, $\overline{R}$, $R_{\min}$, and $\varDelta_{\min}$ were measured in the "steady state", i. e., after an initial transient phase had been passed. A separate simulation was started for every configuration with $B > \mu\tau + 1$. We determined N from (4) or (5) and used it as horizontal axis in the diagram. Furthermore, we normalized the measured values by the theoretical $x_{\max} = C/\mu$.

Figure 8 shows the simulation results in a scatter-plot for a fine-granular timer and a rather small m. The comparison with the analytical results reveals a quite good match, in particular for $R_{\min}$ and $\varDelta_{\min}$. As expected, our analysis underestimates the actual value of $\overline{R}$ when N is large, but only slightly. The largest discrepancies can be observed for $R_{\max}$. As already discussed, $R_{\max}$ highly depends on the shape of the RTT reduction at the beginning of the cycle. Nevertheless, our model is a good approximation for a large parameter range. Besides, it is almost independent of the TCP variant as long as the *Additive Increase Multiplicative Decrease* mechanism is used.

Many TCP implementations use coarse-grained clocks. Unfortunately, a large granularity G introduces noise and degrades the model accuracy. Furthermore, our results are likely to be biased if R takes values below m. The simulation results in Fig. 9 reveal that for $m = 1\,\text{s}$ model and simulation do not match for a number of configurations. Concerning $R_{\max}$ and $R_{\min}$, one can take this effect into account by setting $R'_{\max} = \max(R_{\max}, m)$ and $R'_{\min} = \max(R_{\min}, m)$.

Generally speaking, our analysis applies best if $R(n) > m$ for all n. This, as well as our modeling assumption of a single bottleneck link, is most likely fulfilled by high-latency wireless links, while it may not hold in other networks. We also do not consider jitter, e. g. caused by retransmissions of radio blocks in the link layer. As long as such jitter is on the order of tens of milliseconds, it hardly will trigger spurious timeouts (see next section). But the volatility affects the smoothed variance $v(n)$ and increases $R(n)$. To some extent, one can account for such random fluctuations by replacing μ and τ with their average values $\overline{\mu}$ and $\overline{\tau}$ (see [8]). Our model then provides lower bounds for the RTO duration and therefore is some kind of worst case analysis.

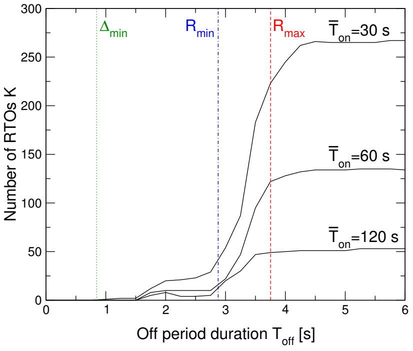

Fig. 10. Number of timeouts for default RTT measurement

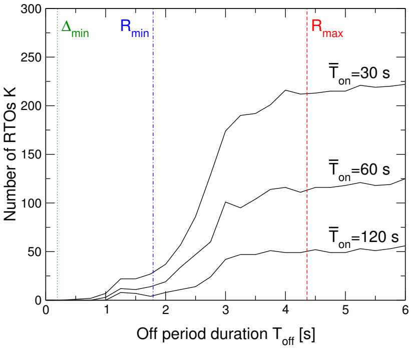

Fig. 11. Number of timeouts for timestamp-based sampling

5 Quantifying the Sensitivity of TCP to Delay Variations

5.1 Spurious Timeout Probability

Based on the knowledge about the duration of the retransmission timeout, we can calculate the probability of spurious timeouts in the presence of delay variations. We assume the uplink and downlink transmission to be interrupted for a duration T_{off}. The time between such events $T_{\text{on}} \gg T_{\text{off}}$ is determined from a negative-exponential distribution. During the "off" period, no packets are transmitted but they remain buffered in the base station and the mobile terminal. The question we are interested in is whether such a delay spike triggers a timeout, which would be spurious because no packets are lost. Taking into account that the retransmission timer is restarted whenever an ACK acknowledges new data [4], this requires the inter-arrival time between two successive ACKs to exceed the current RTO duration. Or, in other words, a timeout should occur if $T_{\text{off}} > R$.

We have verified this by performing simulations for various values of T_{off}. The total number of retransmission timeouts K during a data transfer of about two hours is shown in Fig. 10 and Fig. 11. The configurations correspond to Fig. 6 and Fig. 7, respectively. As a reference, we also added Δ_{min}, R'_{min}, and R'_{max} from our theoretical model, which are calculated assuming no delay variations. This implicitly assumes that T_{on} is longer than the duration of one cycle.

Obviously, K depends on the frequency of delay spikes. The comparison with the total number of interruptions reveals that there is virtually always a (spurious) timeout if T_{off} is large, e. g. six seconds. In contrast, short interruptions do not trigger any timeout. Figure 10 and Fig. 11 show that the critical interval is between R'_{min} and R'_{max}, which are determined from the analytical model. This motivates us to define a *spurious timeout probability* as follows:

$$P_{\text{TO}}(T_{\text{off}}) = \begin{cases} 0, & \text{if} \quad T_{\text{off}} < R'_{\text{min}} \ , \\ \frac{T_{\text{off}} - R'_{\text{min}}}{R'_{\text{max}} - R'_{\text{min}}}, & \text{if} \quad R'_{\text{min}} \leq T_{\text{off}} < R'_{\text{max}} \ , \\ 1, & \text{if} \quad T_{\text{off}} \geq R'_{\text{max}} \ . \end{cases} \tag{19}$$

This formula computes the probability that an interruption of duration T_{off} triggers a timeout. As presented, the parameters R'_{max} and R'_{min} can be determined from the maximum path capacity C and the bottleneck service rate μ, provided that it is known whether the TCP sender uses timestamps or not.

5.2 Evaluation

The *spurious timeout probability* quantifies the risk of inefficient interactions between TCP and delays, e. g. caused by the handover management or by resource allocation mechanism. Nevertheless, recall that R'_{min} is on the order of the maximum RTT x_{max} or m, whichever is larger. Assuming $m = 1\,s$, the data transmission must be interrupted for at least one second to trigger a timeout. Thus, a TCP-friendly system design should avoid such delay spikes. If this is not feasible, one might think about optimization techniques such as transport layer protocol helpers [7, 8] or TCP enhancements like the *Eifel-Algorithm* [5].

Enabling the timestamp option in mobile networks is recommended by [16], claiming that this "reduces the risk of spurious timeouts". The results both of our analysis and our simulation disagree with this statement: Figure 10 and Fig. 11 show that smaller delay spikes trigger more spurious timeouts for timestamp-based measurement. Therefore, one must carefully balance the increased timer aggressiveness against the faster feedback which timestamps are able to provide.

Of course, (19) is only an approximation. A more detailed look at Fig. 10 and Fig. 11 reveals some timeouts for $T_{\text{off}} < R'_{\text{min}}$. They occur for two reasons. First, the retransmission timer is not restarted when no new ACKs arrive, e. g. during the *Fast Recovery*. In this case, a timeout can be triggered by any $T_{\text{Off}} > \Delta_{\text{min}}$. And second, bursty ACK arrival may cause multiple packet drops at the buffer in front of the bottleneck. TCP recovers from this by *real* timeouts. But such buffer overflows are only an issue if the buffer is almost completely filled, and they could be avoided by active queue management techniques.

Finally, it should be mentioned that a spurious TCP timeout may not only be caused by a complete interruption of the data transmission, but also by other delay variations. In particular, a sudden decrease of the service rate from μ_{H} to μ_{L} may have the same effect. An analysis of this scenario in [9] concludes that the retransmission timer will not expire if $R > 1/\mu_{\text{L}}$. With $\mu = \mu_{\text{H}}$ we can determine this value from our model and thus refine this condition: A bandwidth oscillation can only trigger a spurious timeout if $\mu_{\text{L}} < 1/R'_{\text{min}}$.

6 Conclusion

In this paper, we study the characteristics of the TCP retransmission timer for bulk data traffic over low-bandwidth paths. The main contribution is an analytical model to predict the RTO duration from given network parameters. We model both the standard RTT measurement and timestamp-based sampling. The results show that the sampling rate has a significant impact on the timer performance. From this we conclude that an RFC 2988 compliant timer does not harmonize well with timestamps. Given the complexity of underlying TCP mechanisms, we found that our model and simulation results match well over a wide range of scenarios. Based on the theoretical results, we are able to quantify the sensitivity of TCP to sudden delay variations. Our focus are longer off periods. We determine the probability of spurious timeouts depending on the off period duration, the path characteristics, and the TCP parametrization. As a rule of thumb, delay variations are critical whenever they are on the order of one second.

References

1. J. B. Postel (Editor), “Transmission control protocol,” RFC 793, Sept. 1981.
2. A. Gurtov, M. Passoja, O. Aalto, and M. Raitola, “Multilayer protocol tracing in a GPRS network,” in *Proc. IEEE VTC 2002 Fall*, Vancouver, Canada, Sept. 2002.
3. R. Chakravorty, J. Cartwright, and I. Pratt, “Practical experience with TCP over GPRS,” in *Proc. IEEE GLOBECOM*, Taipei, Taiwan, Nov. 2002.
4. V. Paxson and M. Allman, “Computing TCP’s retransmission timer,” RFC 2988, Nov. 2000.
5. R. Ludwig and R. H. Katz, “The Eifel algorithm: Making TCP robust against spurious retransmissions,” *ACM SIGCOMM Computer Communications Review*, vol. 30, no. 1, pp. 30 – 36, 2000.
6. A. Gurtov, “Effect of delays on TCP performance,” in *Proc. IFIP Personal Wireless Communications*, Lappeenranta, Finland, Aug. 2001.
7. J. Schüler, S. Gruhl, T. Schwabe, and M. Schwiegel, “Performance improvements for TCP in mobile networks with high packet delay variations,” in *Proc. 17th Int. Teletraffic Congress*, Salvador, Brazil, Sept. 2001.
8. M. C. Chan and R. Ramjee, “TCP/IP performance over 3G wireless links with rate and delay variation,” in *Proc. ACM MOBICOM*, Atlanta, USA, Sept. 2002.
9. M. Yavuz and F. Khafizov, “TCP over wireless links with variable bandwidth,” in *Proc. IEEE VTC 2002 Fall*, Vancouver, Canada, Sept. 2002.
10. A. A. Abouzeid and S. Roy, “Stochastic modeling of TCP in networks with abrupt delay variations,” *Wireless Networks*, vol. 9, no. 5, pp. 509 – 524, 2003.
11. M. Allman and V. Paxson, “On estimating end-to-end network path properties,” in *Proc. ACM SIGCOMM*, Seattle, USA, Sept. 1999.
12. M. Allman and J. Griner, “TCP behavior in networks with dynamic propagation delay,” in *Proc. IEEE GLOBECOM*, San Francisco, USA, Dec. 2000.
13. T. Lakshman and U. Madhow, “The performance of TCP/IP for networks with high bandwidth-delay products and random loss,” *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336 – 350, June 1997.
14. M. Allman, V. Paxson, and W. Stevens, “TCP congestion control,” RFC 2581, Apr. 1999.
15. V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” RFC 1323, May 1992.
16. H. Inamura, G. Montenegro, R. Ludwig, A. Gurtov, and F. Khafizov, “TCP over second (2.5G) and third (3G) generation wireless networks,” RFC 3481, Feb. 2003.
17. P. Sarolahti and A. Kuznetsov, “Congestion control in Linux TCP,” in *Proc. USENIX Annual Technical Conference*, Monterey, California, USA, June 2002.
18. R. Ludwig and K. Sklower, “The Eifel retransmission timer,” *ACM SIGCOMM Computer Communications Review*, vol. 30, no. 3, pp. 17 – 27, 2000.
19. “IKR Simulation Library,” http://www.ikr.uni-stuttgart.de/IKRSimLib/.