Estimating Frequency and Effects of TCP Spurious Retransmission Timeouts by Traffic Monitoring in Operational GPRS Networks

Francesco Vacirca\textsuperscript{ab}, Thomas Ziegler\textsuperscript{a} and Eduard Hasenleithner\textsuperscript{a}

\textsuperscript{a}ftw. Forschungszentrum Telekommunikation Wien, Donau-City-Strasse 1, A-1220 Vienna, Austria. \{vacirca,ziegler,hasenleithner\}@ftw.at

\textsuperscript{b}INFOCOM Dept. - University of Roma “La Sapienza”, Via Eudossiana 18, 00184, Roma, Italy. vacirca@infocom.uniroma1.it

\textbf{Abstract:} This paper evaluates TCP spurious retransmission timeouts in a large operational GPRS network by post-processing traffic traces captured at Gi interfaces. Applying a novel algorithm to identify spurious retransmission timeouts, the frequency of spurious timeouts as well as the spurious timeout probability dependent on the flow size, the round trip time, and the network load is analyzed. Among other findings, this investigation shows that spurious timeouts are infrequent events in the considered GPRS network. In addition to the evaluation of data from an operational network, testbed measurements are performed to investigate the effect of round trip time variations as seen in the GPRS network on several flavors of TCP. Comparing TCP Newreno, SACK, TCP with Time Stamp enabled, and TCP F-RTO, we can not find major performance improvements in case of advanced TCP derivates. This observation can be addressed to the infrequency of spurious timeouts in scenarios with GPRS-like RTT variations.

\textbf{Keywords:} TCP, Spurious Timeout, GPRS.

1. INTRODUCTION

TCP congestion control continues to be the most important transport layer mechanism enabling stable Internet performance by adapting the transmission rate of flows as a function of the load situation in the network. A great deal of research has been performed in the last years showing that the TCP protocol and its congestion control mechanism can be safely applied to a broad variety of scenarios and transmission media. Due to the migration of the TCP/IP protocol suite onto wireless networks, however, TCP congestion control often has to operate in extreme environments it was not originally designed for. One of the main problems of TCP over wireless networks is its exposure to heavy delay variations happening especially in context of GPRS and UMTS due to sudden increases in the volume of high priority traffic (e.g. GSM voice calls), wireless channel fluctuations, cell handovers, and temporary loss of connectivity. A sudden and extensive increase of the transmission delay in the underlying network obviously causes the Round Trip Time (RTT) at the transport layer to increase steeply, which in turn may cause the TCP retransmission timeout (RTO) to expire. If the retransmission timeout expires, TCP suffers from an ambiguity problem in the
sense that it cannot distinguish whether a packet has actually been lost or packets and their corresponding ACKs merely experienced an extraordinarily high RTT. In the first case, the usual TCP reaction (Go-Back-N and Slowstart) can definitely be considered as appropriate. We will call this kind of RTO a normal RTO (NRTO) for the remainder of the paper. In the second case the RTO has expired unnecessarily, the recovery procedure retransmits up to a full window of data into the network and the congestion window is unnecessarily deflated to one. We will call this kind of RTO a spurious RTO (SRTO) for the remainder of the paper.

Spurious RTOs have recently received significant attention by the research community. Several papers [1–3] propose analytical models of TCP SRTOs in dependency on delay variations. Analytical models are validated by simulation. The Eifel Algorithm [4,5] makes TCP distinguish between SRTOs and NRTOs using the TCP timestamps options. The Duplicate SACK proposal [6,7] aims at detecting SRTOs by an enhancement to the TCP SACK option. This proposal avoids unnecessary reductions of the congestion window but not unnecessary retransmissions in case of SRTOs. The F-RTO modification of TCP error control to detect SRTOs has been proposed in [8]. Compared to Duplicate SACK and Eifel, F-RTO provides the benefit of avoiding interaction between TCP data sender and receiver. [9] proposes a proxy mechanism to detect SRTOs and to filter TCP acknowledgments in order to avoid SRTOs at the data sender. Finally, an analysis of TCP performance over GPRS by passive traffic monitoring at the Gi interface is performed in [10].

Reading above paragraphs we observe that research so far has concentrated on analytical modeling of SRTOs and the improvement of TCP to cope better with SRTOs. What is missing, however, is an extensive investigation on the actual frequency of SRTOs in today’s operational wireless networks and, based on this, a quantitative analysis of the effect of SRTOs in degrading TCP performance in a realistic scenario. The research motivated above has been made possible due to the performance of extensive traffic monitoring in the core of a large operational GPRS network.

Overview of the paper: the GPRS monitoring infrastructure and the testbed scenario are described in Section 2. In Section 3 we illustrate an algorithm to distinguish SRTOs from NRTOs. In Section 4 we use the algorithm to investigate the frequency of SRTOs in the operational network based on two days of packet traces taken at the Gi interfaces of two different GGSNs. Section 5 compares standard TCP flavors (NewReno, SACK and Time Stamp) with TCP having the F-RTO algorithm enabled, to investigate the performance improvement using infinite length TCP flows as well as Web-like TCP traffic. Finally, in Section 6, we resume the main conclusions of the work and give some hints for further work.

2. MONITORING AND TESTBED INFRASTRUCTURE

This paper uses an extensive traffic monitoring infrastructure in the operational GPRS network to retrieve packet traces. The monitoring infrastructure is composed of high-end PCs equipped with Endace DAG network monitoring interface cards [11] that have been installed at several Gi and Gn interfaces in the GPRS network. The monitored network is the large Mobilkom Austria GSM/GPRS network having more than 3 Million customers. For reasons of the provider's privacy we omit more detailed information on the architecture of the GPRS core network. Captured packet traces are anonymized in realtime for privacy.

\[1\] The setup of the measurement infrastructure has been accomplished in cooperation between Kapsch Carrier Com, Mobilkom Austria, and ftw. within the Kplus project META WIN (MEasurement and Traffic Analysis in Wireless Networks). The research for this paper has been performed within META WIN and the project N0 which is funded by the Austrian Kplus research program and industrial partners of ftw.
reasons and include IP as well as TCP headers enabling the investigation of all kind of
TCP related statistics using a variety of scripts which have been implemented. Traces are
stored on high capacity removable hard disks which are transferred to the research lab where
statistical evaluations take place. In the context of this paper whole-day traces from two
different Gi interfaces are used. These traces were taken on Thursday May 6th, and Friday
May 11th 2004. We use tcptrace [12] standard features to derive GPRS network and

![Semi Round Trip Time (ms)](image)

Figure 1. Experimental CDF of semi RTT.

<table>
<thead>
<tr>
<th></th>
<th>SACK</th>
<th>TS</th>
<th>DSACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 6th, Gi1</td>
<td>83.64%</td>
<td>18.84%</td>
<td>1.8%</td>
</tr>
<tr>
<td>May 6th, Gi2</td>
<td>83.35%</td>
<td>15.19%</td>
<td>1.86%</td>
</tr>
<tr>
<td>May 11th, Gi1</td>
<td>81.35%</td>
<td>13.96%</td>
<td>1.88%</td>
</tr>
<tr>
<td>May 11th, Gi2</td>
<td>83.86%</td>
<td>16.87%</td>
<td>2.48%</td>
</tr>
</tbody>
</table>

Table 1. TCP flavor statistics.

traffic characteristics that are then exploited to set up the laboratory testbed described
at the end of this section. The first characteristic derived is the Semi RTT Cumulative
Distribution Function (CDF). Semi RTTs can be seen as the RTT at the TCP layer between
the mobile node and the Gi interface. Investigating the RTT of GPRS traces we observe that
the RTT variations between the Gi interface and nodes in the wired Internet are negligible
compared to the semi RTTs between the mobile node and the Gi interface. Thus using semi
RTTs instead of full RTTs suffices to reproduce realistic RTT variations in our testbed. The
Cumulative Distribution Functions of the captured semi RTTs from the May 6th and 11th
traces are shown in Figure 1. We observe that the CDFs of RTTs derived from different days
and different interfaces are similar. The second issue we address is the percentage of TCP
connections that utilizes TCP options. Table 1 shows the percentage of TCP connections that
utilize the SACK option, the Time Stamp (TS) option and the Duplicate SACK (DSACK)
option in the analyzed GPRS traces; the sets of TCP connections that have the SACK,
the TS and the DSACK enabled are not disjoint. It is to notice that most of the TCP
connections (more than 80%) in the GPRS network enable the SACK option, the TS option
is quite widespread (between 14 and 19%), whereas the DSACK option is not so common
(about 2%). For these reasons, in Section 5 we use the SACK and TS options to compare
the performance of various TCP versions. As far as other TCP flavors are concerned, it
is not possible (by passive monitoring) to derive the spread of other options that are not
negotiated explicitly during the setup phase of TCP connections (e.g. F-RTO).

In addition to the monitoring infrastructure a laboratory testbed with a delay emulator
and realistic traffic generators are used to evaluate the performance of different TCP flavors
in the presence of SRTO events. The testbed is composed of two PCs (server hosts) for traffic
generation and two PCs (client hosts) for reception of traffic. Hosts run the Linux OS and are connected to two Ethernet switches with full duplex 100BaseTX links. The two switches are connected to a delay emulator PC with 1000BaseSX links. The delay emulator PC [18] is equipped with SysKonnect NICs and operated under realtime Linux for exact reproduction of propagation delays and delay variations. The purpose of the delay emulator is to reproduce the delay variations typically existing in wireless networks in the testbed. This is required to apply realistic RTT variations to TCP flows when evaluating the performance of different TCP flavors by testbed measurements; to this aim the delay emulator is fed with GPRS semi-RTT traces discussed previously. Using an optical splitter a monitoring host is connected to the link between the left hand side switch and the delay emulator. The monitoring host is equipped with a Gigabit Ethernet DAG card and a harddisk to store traffic traces.

The traffic is generated by a FTP-like traffic generator and a web traffic generator. For FTP-like traffic (infinite length flows), we use tcpc [15] to generate TCP flows from the servers to the clients; for web traffic we use the websim utility [19] that generates traffic according to the Scalable URL Reference GEnerator (SURGE) model [16]. This utility is divided into a multithreaded server application which listens for connection requests and a client application which sends the requests for pages and objects. For managing parallel TCP connections at the client side a thread is spawned for every connection; the client and server software contains logic for efficiently generating load representing several web users per host. SURGE random variables implemented in websim are the user think time (Pareto distributed), file size (combined Lognormal and Pareto distribution), inter object time (Weibull distributed), and the number of objects per page (Pareto distributed).

3. SRTO DETECTION ALGORITHM

We exploit an algorithm designed to identify the spurious timeout event probability by passive monitoring that has been originally proposed in [17]. The algorithm is able to discriminate between a normal timeout retransmission (due to packet losses) and a spurious timeout retransmission by exploiting the information contained in the ACK flow received by the monitoring interface before and after the retransmitted packet. It is based on investigating only essential operational principles of TCP in standard situations and it does not exploit TCP options or specific properties of TCP derivates in order to be able to cope with all TCP implementations. A detailed description of the algorithm and of situations where the algorithm is prone to errors as well as an investigation of the algorithm’s accuracy can be found in [17]. In order to achieve a better understanding of the algorithm’s operation we illustrate some NRTO and SRTO examples by investigating short snapshots of real GPRS traces.

![Figure 2. Normal timeout example.](image1)

![Figure 3. Ambiguous timeout example.](image2)
Figure 4. Spurious timeout example.

Figure 2 shows an example of a NRTO extracted from the GPRS traces. The TCP flow has a window of 3 segments; segment 73508 and segment 74888 are lost after they have been captured by the monitoring interface (dashed line). If we focus on the ACKs crossing the monitoring interface before the retransmission of segment 73508 it is easy to identify this retransmission as a retransmission due to NRTO because a duplicate ACK for the lost packet is received indicating a hole in the data receiver’s sequence number space. The second normal timeout example (Figure 3) artificially modifies the trace snapshot shown in Figure 2 by dramatically increasing the RTT of the duplicated ACK 73508 such that it arrives at the monitoring interface after the retransmission of segment 73508. Figure 4 depicts the SRTO example. By observing the ACKs received before the retransmission of the lost packet it is impossible to discriminate between a normal timeout and a spurious one. Comparing Figure 3 with Figure 4, it can be noticed that there is no difference in the sequence of segments and ACKs seen by the monitoring interface between the first transmission of segment 73508 and its retransmission. The two situations can be discriminated only by observing the ACKs seen after the retransmission at the monitoring interface: in case of packet loss we expect to see a duplicate ACK for the lost segment, whereas in the spurious timeout case we expect to see on or more ACKs acknowledging sequence numbers higher than the retransmitted segment. The algorithm exploits these informations to identify the cause of the timeout event\(^2\).

The operational principle of the algorithm can be explained as follows: if a segment is seen twice by the monitoring interface, the monitoring software enters a retransmission state. The considered retransmission states are three: i) fast retransmission state (FRTX), ii) normal timeout retransmission state (NRTO) and iii) spurious timeout retransmission state (SRTO). The state depends on the number of duplicate ACKs seen for the packet before the retransmission. If three or more duplicate ACKs have been received the algorithm supposes that the retransmission is due to a fast retransmit (it enters the FRTX state). If one or two duplicate ACKs are seen, it enters the NRTO state and if zero duplicate ACKs are captured the algorithm enters the SRTO state. In the last case, the algorithm waits for the next ACKs to check if the retransmission is really spurious or not. Strictly speaking, if duplicate ACKs are seen after the reception of the retransmission, the algorithm can be sure that a packet with a sequence number greater than the retransmitted one has been received by the data receiver. Assuming no packet misordering in the network this happens only if the segment has been lost. In this case the algorithm moves to the NRTO state, otherwise it remains in the SRTO state. The algorithm gets out of a retransmission state when the ACK with the highest outstanding sequence number is seen (recovery ACK). This prevents the algorithm from falsely detecting the retransmissions following a fast retransmit event.

\(^2\)The algorithm has been implemented as an optional feature of the tcptrace. The modified version of the code can be downloaded from http://userver.fw.at/~vacirca/
or a normal timeout retransmission as retransmissions due to a SRTO, and vice versa. If the retransmitted segment is the highest segment sent so far the algorithm does not enter any retransmission state but solely marks the retransmission as ambiguous because it is not possible to identify the retransmission cause without exploiting more informations.

4. ESTIMATING SRTOs IN THE GPRS NETWORK

The main issue we address is to understand what is the occurrence of SRTOs in an operational GPRS network and how these events are correlated to TCP connection characteristics. Table 2 reports the percentage of TCP connections with a number of packets less than 10, between 10 and 100, between 100 and 1000 and greater than 1000. These four different TCP connection categories are further divided according to the number of SRTOs experienced on a per connection basis; in particular they are divided in the connections that experienced zero SRTOs, one SRTO and more than one SRTO. From the table it is clear that i) most of the TCP connections carry few packets: approximately 56% of connections carry less than 10 TCP packets, 98% less than 100 packets and less than 2% of connections carry more than 100 packets. Of all the connections, only 1.2% experience a single spurious timeout and 0.22% experience more than one spurious timeout. As shown in the table, spurious timeouts are more frequent for connections with a larger number of transmitted packets, e.g. connections with more than 1 SRTO that carry more than 1000 packets are 0.046%. On the contrary, we observed no connections with less than 10 packets and more than 1 SRTO.

Figure 5 shows a per connection scatterplot of the mean RTT (upper plot), the standard deviation of the RTT (middle plot) and the number of packets (lower plot) with respect to the number of SRTOs experienced by the connection. The solid line represents the average of the mean RTT, RTT standard deviation and packets per connection, respectively. Moreover 95% confidence intervals of the average values are depicted showing the statistical relevance of the results. As far as the number of packets per connection is concerned, the increasing trend of the solid line in the lower part of the figure shows a clear positive correlation between the number of SRTOs per connection and the number of packets transferred during the duration of the connection. The upper and the middle plots show an evident correlation between the connections that do not experience any SRTO and the connections that have a low mean RTT and a low RTT standard deviation. Moreover, considering only the connections that

<table>
<thead>
<tr>
<th>Packets per Connection</th>
<th>Overall</th>
<th>0 SRTO</th>
<th>1 SRTO</th>
<th>&gt; 1 SRTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x &lt; 10$</td>
<td>55.815%</td>
<td>55.79%</td>
<td>0.025%</td>
<td>0%</td>
</tr>
<tr>
<td>$10 \leq x &lt; 100$</td>
<td>42.589%</td>
<td>41.352%</td>
<td>1.158%</td>
<td>0.079%</td>
</tr>
<tr>
<td>$100 \leq x &lt; 1000$</td>
<td>1.529%</td>
<td>1.235%</td>
<td>0.198%</td>
<td>0.096%</td>
</tr>
<tr>
<td>$x \geq 1000$</td>
<td>0.067%</td>
<td>0%</td>
<td>0.021%</td>
<td>0.046%</td>
</tr>
</tbody>
</table>

Table 2
Percentage of TCP connections dependent on number of packets and SRTOs.
experience one or more SRTOs, the correlation between the mean and the standard deviation of the RTT and the SRTO seems to be a negligible effect as revealed by the almost constant trend of the solid line of the upper and middle plots for one or more SRTOs.

The last issue we point out from the GPRS traces, is the effect of the network load on the SRTO events. In all three subplots of Figure 6 the x axis represents the time of day in units of hours. The upper part figure shows overall number of SRTO events divided by the number of packets seen in a time interval of two hours. The middle part figure shows the overall number of SRTO events divided by the number of congestion recovery events performed by TCP, i.e. the number of SRTOs plus NRTOs plus fast retransmits and ambiguous retransmissions. The bottom part figure shows the normalized out-of-order probability during the two monitored days and at the two different Gi interfaces. Every point of the plots is the average of a two hour trace. A plain time of day effect of the out-of-order probability is visible, this trend reflects the load situation of the network during the different hours of the night and the day. The time of day effect is not present in the spurious timeout probability that maintains almost constant and it is uncorrelated to the out-of-order probability (i.e. the load of the network). That leads to the conclusion that the delay variations in the GPRS network are caused by phenomena (e.g. fluctuations of the radio channel and increasing of GSM traffic) that are independent of the network load. Moreover, analyzing the upper and the middle plot of the figure, it is possible to observe that on average only one packet over one thousand experiences a spurious timeout and that these events are only a small fraction of the number of congestion recovery events performed by TCP (SRTOs, NRTOs, fast retransmits and ambiguous retransmissions). Leaving aside the
outlier point reflecting the Gi2 interface in the two monitored days at 2 a.m., the maximum value of the ratio between the SRTOs and the congestion events is of the order of one over twenty. That implies that the performance degradation due to SRTOs is low because other congestion recovery events prevail over the SRTO events.

5. TCP PERFORMANCE EVALUATION

In this section we investigate the effectiveness of different TCP flavors with respect to TCP spurious timeouts using the testbed described in Section 2. The TCP derivates compared in the analysis are the Linux version of NewReno, SACK, Time Stamp and F-RTO; as shown in Section 2, SACK and TS options are frequently applied in the GPRS network scenario.

![Graph showing retransmissions per spurious timeout varying with the number of FTP connections for different TCP flavors.](image)

Figure 7. Average number of packet retransmissions per spurious timeout varying the number of FTP connections for different TCP flavors.

Figure 7 depicts the average number of retransmissions that occur after a spurious timeout expiration for different versions of TCP. It is possible to observe that when the network load is low F-RTO reacts well to the spurious timeout events and it avoids useless retransmissions. When the traffic load increases, the TCP congestion window becomes smaller reducing the number of unnecessary retransmissions after the timeout and thus the number of retransmitted packets for different versions of TCP converges to the same value. Figures 8(a) and 8(b) depict the TCP throughput in the FTP scenario and the throughput per object in the Web scenario, respectively. The first figure shows that there are no big differences between the performance of different TCP flavors; however it is to notice that the SACK option combined with the F-RTO recovery procedure achieves the maximum throughput for every value of the load and that F-RTO alone obtains the worst performance. In case of Web traffic (Figure 8(b)) the results show that TCP SACK with F-RTO is efficient only if the number of users (i.e. the load) is high. If the number of users is low, the best performance is obtained by the TS option.

Summarizing above observations, we find that F-RTO minimizes the number of retransmitted packets during the recovery phase after a SRTO retransmission leading to a capacity saving in low load situations. However, in no scenario we can observe performance improvements in
terms of throughput of F-RTO compared to other TCP flavors; this is consistent considering the low SRTO probability for GPRS like RTT variations described in the previous section.

6. CONCLUSIONS

In this paper, an algorithm for identification of spurious TCP retransmission timeouts by passive monitoring is applied to real GPRS packet traces and to packet traces captured in a laboratory testbed. Exploiting packet traces captured on an operational GPRS network, we quantify the dependency of the spurious timeout probability on the size of TCP flows, the mean RTT and the standard deviation of a TCP flow’s RTT. Additionally, investigating the spurious timeout probability in dependency on the network load, we find that spurious timeout events are infrequent compared to other events causing a reduction of the congestion window. Furthermore, the probability that a packet experiences a SRTO is small.

By laboratory testbed measurements we compare the performance of several versions of TCP, including a TCP derivate specifically designed to avoid the potentially malicious effects of spurious timeouts, in scenarios with heavy RTT variations. FTP-like and Web-like traffic generators are used in combination with a delay emulator fed with realistic RTT traces from the GPRS network. Not surprisingly considering the main finding of the last paragraph, we can not observe significant throughput improvements of the TCP derivate designed for spurious timeout avoidance compared to standard versions of TCP.

Thus, given the infrequency of spurious timeouts in the monitored operational GPRS network and the resulting non-effect on the performance of the TCP derivate designed to avoid the malicious effects of spurious timeouts, this paper gives strong indications that there exists no need to upgrade standard TCP to improve performance over GPRS networks. In our future work we plan to extend this analysis to UMTS networks in order to evaluate whether the above statement equally holds for 3G networks. Additionally, based on traces collected at Gm and Gb/Iu interfaces, we intend to perform a quantitative analysis of the causes for RTT variations in 2.5G and 3G networks.
REFERENCES

17. F. Vacirca, T. Ziegler, E. Hasenleithner, Large Scale Estimation of TCP Spurious Time-out Events in Operational GPRS Networks, COST-279 TD(05)003.