A Large-scale, Passive Analysis of End-to-End TCP Performance over GPRS

Peter Benko, Gabor Malicsko and Andras Veres
Traffic Analysis and Network Performance Laboratory (TrafficLab)
Ericsson Research
Budapest, Hungary
{Peter.Benko, Gabor.Malicsko, Andras.Veres}@ericsson.com

***Abstract*—In this paper a passive methodology for TCP performance evaluation over General Packet Radio Service (GPRS) networks is presented that relies on traffic monitoring at the GPRS ingress/egress router interface (Gi). Based on the IP and TCP headers of the packets we estimate the end-to-end performance of TCP connections such as connection setup behavior and data transfer goodput. In order to identify the effects behind the measured performance the introduced algorithms estimate round trip delays, packet loss ratios, available channel rates, throughput and carry out bottleneck analysis. Large-scale GPRS measurements in seven countries are presented to analyze TCP performance and demonstrate the applicability of the method. The effects of different TCP parameters such as maximum segment size, selective acknowledgements, timestamp usage and receiver window size are also quantified. GPRS measurement results are compared to a wireline dial-up network to identify the effects specific to the wireless environment.**

Keywords-network measurements; GPRS; TCP; passive measurement; performance evaluation; wireless networks

I. INTRODUCTION

During the last years, the General Packet Radio Service (GPRS) [1] has taken off all over the world. Today, the number of operational or planned networks providing GPRS service exceeds one hundred.

Because of the fundamental differences between the wireless and the traditional wireline architectures, protocols and environments, the real-life performance of Internet access over GPRS has raised considerable attention. A number of research groups have performed measurement studies to investigate the quality of service in GPRS networks. The emphasis was put on determining TCP's tolerance to the increased round-trip delays, fluctuating channel rates and special loss patterns experienced in the wireless environment. Based on the measurements, several factors and bottlenecks have been identified that influence the performance of TCP. Suggestions have also been made to improve TCP, e.g., by manipulating the sending window [2][3], or by modifying the TCP congestion control and loss recovery algorithms [4][5][6]. GPRS specific optimizations have also been introduced [7][28].

Previous research focused on simulations or active measurements in selected and usually well-controlled scenarios [27][20][2][3][18]. During the studies, the researchers had to choose a realistic measurement setup that they believed to be representative. The choice had to consider the possible set of terminals (3+1, 4+2 timeslot capabilities), cell areas (urban, rural), TCP flavors (Reno, Selective Acknowledgements – SACK), operating systems, mobility patterns (stationary, highway), applications and much more.

GPRS networks are more inhomogeneous in nature than wired networks. We argue that the measurement setup poses a huge multidimensional problem, which cannot be solved sufficiently by using selective active tests or simulations. In this paper, we chose a fundamentally different approach to this problem. Instead of investigating certain arbitrary setups, we focused on the actual use cases realized by hundreds of thousands of GPRS subscribers in real life. This method is based on a passive analysis of packet traces obtained on the Gi interface [1]. We collected packet traces from operational GPRS networks in seven countries. The duration of measurements varied between a few days to four weeks each. The summary statistics and the measurement configuration are described in detail in Section II.

In this paper, we introduce a new set of algorithms that enable the analysis of TCP traffic traces obtained from this middle point (Gi) in the connection path. The algorithms focus on finding the main bottlenecks for end-to-end TCP data transfer following the critical path analysis philosophy introduced in [17]. In particular, in Section III, we present a bottleneck analysis of TCP connection setup procedure. We define a TCP connection setup "performance tree", which classifies the observed TCP connection setup protocol sequences as branches, and quantifies the weights based on GPRS and wireline modem measurements.

In GPRS networks, the available capacity is influenced by several factors, such as voice and GPRS data load, as well as the terminal timeslot capability. In Section IV, we define a new term called GPRS channel rate, which is the maximum achievable IP layer throughput on the GPRS channel. We present an algorithm that estimates the channel rate value from the packet traces.

Packet loss is estimated using a passive algorithm presented in [12]. We compare GPRS with the dial-up network and quantify the probability that the bottleneck lies in the GPRS access or in the public Internet. The results can be found in Section V.

Measuring TCP throughput is quite straightforward using active measurements, but it is challenging using passive methods. On one hand, the problem is that users usually transfer several files at the same time. On the other hand, the monitoring point is located before the bottleneck buffer. Due to these problems, an analysis of individual TCP connections would lead to a wrong conclusion. We present an algorithm that takes the parallel traffic into consideration and accounts for the effects of the bottleneck buffer at the same time. The algorithm and measurement results are presented in Section VI.

Recently, a large amount of research effort has been put into improving TCP's data transfer performance especially in wireless systems. However, there are no papers yet that study the effects of the proposed improvements based on large statistical basis in GPRS. In Section VII, we quantify the performance improvements of some basic TCP parameters over GPRS, such as having different Maximum Segment Sizes (MSS) and window sizes, and using Selective Acknowledgements (SACK) or TCP timestamps.

II. MEASUREMENT SETUP

The data collection and analysis device was installed in the GPRS Core Network on the Gi GPRS interface as shown in Fig. 1. All downlink and uplink user IP data flows through this interface. The Gi interface is the borderline between the GPRS domain and the Internet. The Gateway GPRS Support Node (GGSN), just before the Gi interface terminates all mobile network specific protocols.

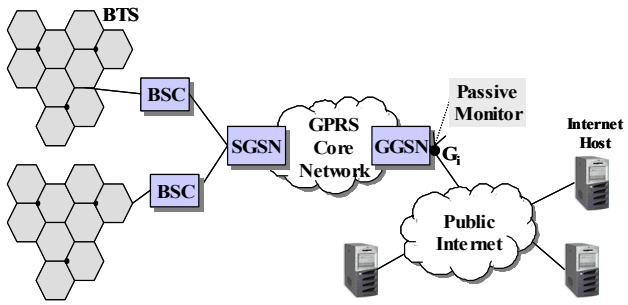

Figure 1. GPRS measurement configuration

The packet traces were collected in seven European and Asian countries. The duration of the measurements varied between a few days to four weeks in each case to capture statistically significant number of samples. The summary statistics of the collected data is shown in Table I. Over 10 million TCP flows were observed, which enables reliable statistical analysis. The rows in Table I contain the following information.

Number of TCP flows stands for the overall number of TCP connections captured. It includes both successfully set up and broken TCP connections (e.g., we count SYN-RST packet pairs as broken TCP connections). We only consider connections that were initiated by the mobile host. This is the typical case (with only very few exceptions) in current GPRS access networks.

SACK enabled specifies the ratio of TCP Selective Acknowledgement (SACK) [9] enabled TCP connections. We assume that if the SACK permitted option was seen for both ends at connection setup time, the endpoints were capable of generating selective acknowledgements during the lifetime of the connection.

The *TCP timestamp usage* column shows the ratio of TCP connections for which the TCP timestamp option [10] was seen in both directions.

MSS low and *high* specify the ratio of TCP connections that used a "low" or "high" Maximum Segment Size (MSS). The MSS was determined from the maximum size of the IP packets seen on the TCP connection, which is 400-600 bytes for *MSS low* and 1400-1600 bytes for *MSS high*.

Wnd low, *med* and *high* show the ratio of TCP connections where the maximum of the advertised receiver window size in the GPRS terminal was "low", "medium" or "high". A window size below 10 Kbytes was considered *low*, between 10 Kbytes and 20 Kbytes as *med* and beyond 20 Kbytes as *high*.

TABLE I. SUMMARY STATISTICS

	Sum of all GPRS	**Wireline dial-up**
# TCP flows	12,587,056	225,932
SACK enabled	55.49%	59.48%
TCP timestamp	9.48%	2.45%
MSS low	25.12%	49.29%
MSS high	22.52%	7.96%
Wnd low	55.90%	96.74%
Wnd med	25.54%	1.01%
Wnd high	18.56%	2.25%

It can be seen that more than half of the TCP connections is SACK enabled. Interestingly, the TCP timestamp option is not so widespread, only less than 10% of the connections actually use it. The reason behind this is that although current operating systems (Linux, Windows 9x, XP, 2000, etc.) support TCP timestamps, they are turned off by default for active TCP opens [22]. That is, the initiator of the TCP connection usually does not request the timestamp option in its SYN segment. It is worth noting that this behavior does not influence the possible deployment of TCP Eifel, which is proposed to enhance TCP performance over wireless networks [5]. An Eifel enabled host may request timestamps when initiating a TCP connection, thus forcing its peer to use timestamps even if they are disabled by default.

Table I also shows that low (400-600 bytes) and high (1400-1600) MSSs are nearly equally used in GPRS. This is not true for the maximum receiver window sizes in the GPRS terminals, since it is less than 10 Kbytes for more than 50% of the TCP connections.

III. TCP CONNECTION SETUP PERFORMANCE

We split the TCP analysis into two main parts: TCP connection setup and data transfer. Because of the relatively high round-trip delay in GPRS, the TCP connection setup takes considerably longer than in wired networks. In addition, since TCP relies on timers during the three-way handshake procedure, it may be sensitive to long delays and occasional packet losses introduced by the wireless access. In this section, we analyze the real-life impact of GPRS characteristics on the TCP connection setup procedure.

The connection setup phase is split further to several sub-categories depending on whether the setup was successful or not, and on the major state transition patterns we could observe in the traces.

A. TCP connection setup delay measurement

The connection setup delay is calculated as the time difference between the first SYN packet and the first ACK packet seen (for an example, see Fig. 2). Note that this delay is slightly different from the delay that the mobile host experiences.

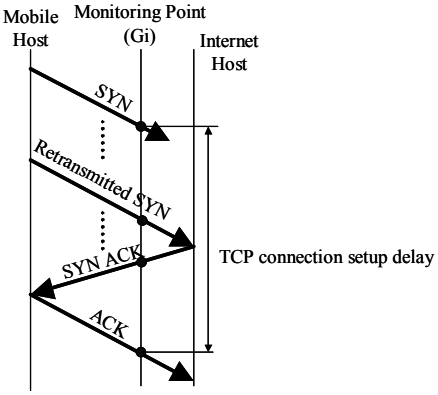

Figure 2. TCP connection setup delay

The histogram of the overall setup delay in a typical GPRS network is shown in the top sub-graph of Fig. 3. The setup delay has a peak around one second, and it has a quite long tail reaching delays as large as 5-10 seconds. This delay distribution is a result of several factors. We break down the delay to conditional delays with different probabilities according to the following considerations.

The optimal case with the lowest setup delay is when no SYN retransmissions are seen (*1 SYN, 1 SYN ACK*). The delay may be increased due to SYN packet retransmissions (*1+ SYN and/or 1+ SYN ACK*) caused by losses or too large delays.

The *background traffic* and *no background traffic* categories are used to differentiate between the cases when (i) the connection setup packets are likely to be mixed with packets from other connections from the same user and (ii) when the bottleneck buffer is empty and no buffering delay is suffered. In order to be sure that the bottleneck buffer in the GPRS network is flushed, a silent period of at least 5 seconds is required before the connection setup in the *no background traffic* case.

The combination of the above considerations results in four possible connection setup categories. All traced TCP connections fit into one of these four categories. See Fig. 3.

It can be seen in Fig. 3, that the head of the overall distribution is dominated by the *1 SYN, 1 SYN ACK* cases. Additionally, if there is no background traffic, the delay is almost deterministic with a short tail. The tail of the overall distribution is dominated by the category where background traffic is observed, which means that self-congestion is the most significant bottleneck in this respect. The last two cases: *1+SYN and/or 1+SYN ACK* causes high delays in setup time too, but the probability of falling into this case is rather low.

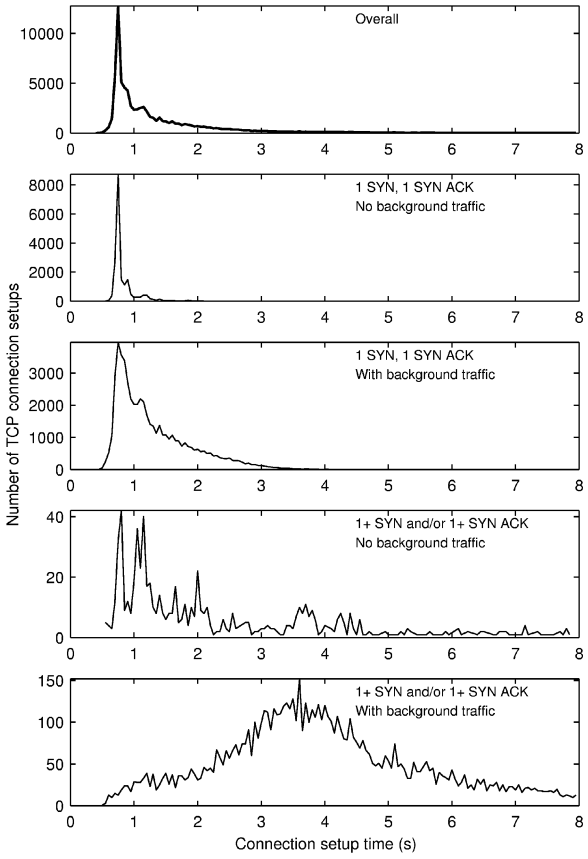

Figure 3. Connection setup time components in a GPRS network

B. Round-trip time measurement

The GPRS side round-trip delay between the mobile host and the GGSN is determined by a number of factors. It depends on the available capacity in terms of the number of timeslots allocated to the mobile, and on the number of Radio Link Control (RLC) layer [1] retransmissions necessary to successfully deliver the packets uplink and downlink. The time it takes to assign resources at the air interface by setting up Transmission Block Flows (TBF) [1] is also a significant contributor. It has to be noted that in addition to GPRS data traffic volume the voice traffic has an impact on the GPRS quality of service. It is an important question whether daily fluctuations in GPRS service quality exist or not.

Let us revisit the special case when there is no background traffic and there is just one SYN and one SYN ACK packet seen (Fig. 3b). Note that in reality the observed number of SYN or SYN ACK packets may be different from the number of actually sent packets because retransmitted packets might be lost before they reach the monitoring point. Nevertheless, in this special case, it is known for sure that the SYN ACK must be the response to the seen SYN and not to any other lost packet. Similarly, the last ACK packet of the connection setup must have been sent in response to the SYN ACK captured before. This property enables us to estimate both the Internet side round-trip delay (SYN to SYN ACK) and the GPRS side round-trip delay (SYN ACK to ACK) with good precision. There is some error caused by larger delays than the TCP initial retransmission timeout (RTO, recommended value is 3 seconds [11]). Due to this, delays greater than 3 seconds cannot be measured this way.

The cumulative distribution function (CDF) of the measured round-trip delay in the GPRS side is depicted in Fig. 4. The peak-hour (10-12 h) delay and the off-peak (3-7 h) are shown separately. The peak-hour delay is higher than the off-peak delay, but this difference is minor and causes a 50-100 ms shift in the distribution.

For comparison, the delay in the dial-up network is also shown. The dial-up delay compared to GPRS is not only smaller (between 100-300 ms), but its variance is also lower. In addition, GPRS delay has some typical values unlike dial-up access.

C. *TCP connection setup bottleneck analysis*

The procedure used to classify TCP setup delays can be extended to include failed connection setup attempts, which makes it possible to construct a "connection setup classification tree" as shown in Fig. 5. This method allows tracing the connection setup procedure and helps to identify performance problems and their possible causes.

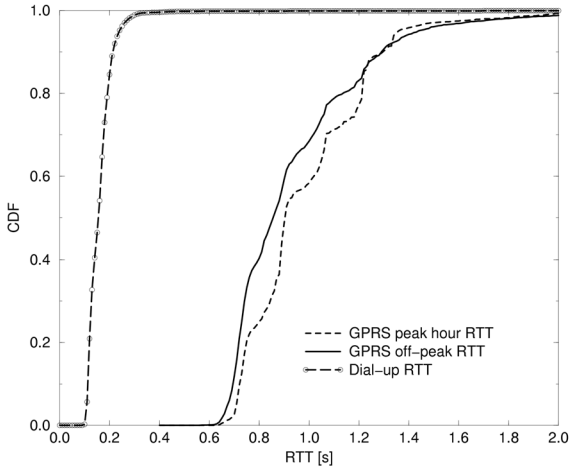

Figure 4. CDF of GPRS and dial-up round-trip time

In Fig. 5, we compare the results of the GPRS measurements (denoted with G) with the dial-up modem network (denoted with M). For each class in the classification tree we specify the overall ratio of TCP connections that fall into the given category (in circles). Each node on the tree contains information on selected metrics for the class i.e. the average setup delay and the ratio of connections that experienced background traffic in the setup phase.

From a performance evaluation point of view, the tree has four main areas of interest. In the *Optimal Area* only a single

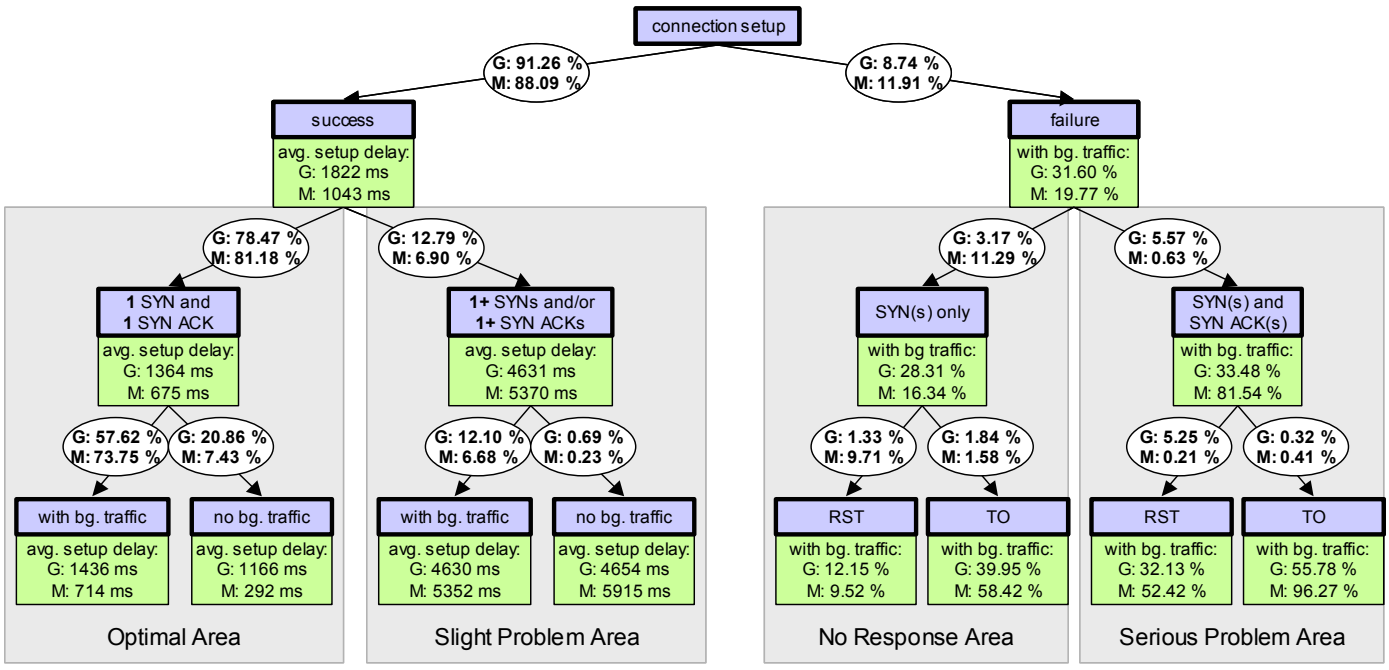

Figure 5. TCP connection setup classification tree

SYN and its corresponding SYN ACK is seen. This is the optimal case.

The *Slight Problem Area* denotes the cases when at least one retransmitted SYN or SYN ACK packet is seen because of packet losses or increased delays. If there is background traffic from the same user, the increased delays experienced by TCP setup packets can be explained by large queuing delays in the bottleneck buffer. There are several proposals to decrease queuing delays in slow access networks and especially in GPRS. [28] proposes to use a channel adaptive RED algorithm in the bottleneck buffer, which tries to ensure a delay target over the varying rate channel. The authors of [2] and [3] suggest to manipulate the receiver window size in TCP acknowledgement packets to limit the amount of outstanding data. It is worth noting that SYN retransmissions indicate that the TCP initial retransmission timeout value may be too low to tolerate the large delays in GPRS.

In the *No Response Area* only one or more SYNs but no corresponding SYN ACK segments are seen. This often occurs when someone tries to connect to a host that is currently not online or is not listening on the specified port. Digging deeper in the traces we found that besides port scanning some peer-to-peer file sharing applications also tend to generate such traffic patterns.

In the *Serious Problem Area* both SYN(s) and corresponding SYN ACK(s) are seen. The SYN ACK indicates that the destination host is willing to accept the connection but the initiator gives up and does not send the ACK. This may happen if excessive delays or severe packet losses are present in the network. If the connection terminates with a RST segment, it is very likely that the user aborted the setup procedure explicitly because of the large waiting time.

Comparing the results of GPRS with dial-up network measurements, it can be seen that the overall setup time is nearly 2 s, which is almost the double of the dial-up setup time. In GPRS, the best-case setup time is more than 1 s while it is around 300 ms in the dial-up case (Optimal Area, no background traffic). However, in the dial-up scenario, only 7% of the TCPs experience the lowest delay in contrast to the more than 20% over GPRS.

IV. Available Channel Rate Estimation

In order to have a basis for the TCP data transfer performance investigations, we have to have a picture of what the maximum achievable IP layer throughput on the GPRS channel is. For this purpose, we have constructed an algorithm that estimates the available downlink GPRS channel rate for a TCP connection.

The goal of the algorithm is to identify periods in time when the channel is fully utilized. During a TCP connection, such periods are most likely when TCP is in its "normal" operating state. That is, there are no packet losses or out-of-order transmissions (i.e. TCP is not in back-off, fast-retransmit or time-out). Additionally, during this period the amount of unacknowledged outstanding TCP data (flightsize) should be "high enough" to keep the bottleneck buffer saturated. The limit should be set in a way that it exceeds the expected bandwidth-delay product of the end-to-end path. In our study, this threshold was set to 4500 bytes.

The algorithm is capable of estimating the bottleneck rate if the bottleneck buffer is located after the measurement point. This is true for the downlink direction in our case. Fig. 6 illustrates the time periods when channel rate measurements can be made on a TCP.

In order to calculate the downlink channel rate, we have to summarize the *total* amount of IP data that arrives to the mobile during the measurement period and divide it with the time it took. In order to do that, the algorithm keeps track of the amount of IP data sent towards the mobile between the start and the end of the measurement period. When the acknowledgement for the last TCP segment of the measurement period arrives, we assume that all the data have arrived to the mobile. The time is calculated from the first and the last acknowledgement packets (t_{end}-t_{start}).

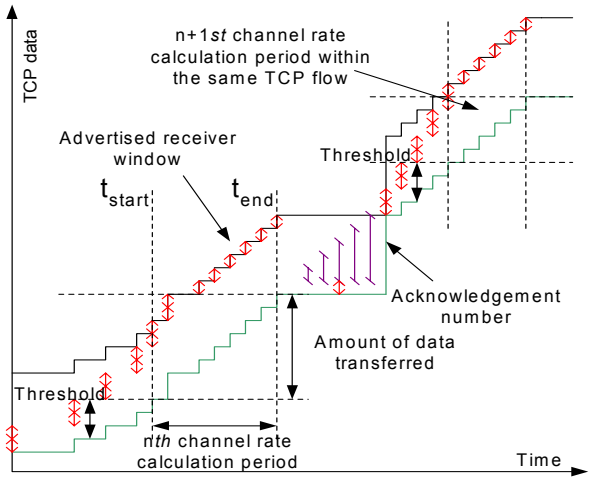

Figure 6. Channel rate estimation

It has to be noted that the jitter in the acknowledgement transmission times may distort the results. To compensate for this jitter we set a lower bound on the transmitted data during the measurement period. If the amount of data does not exceed 16 Kbytes, we do not calculate the channel rate for that period. Another source of error is that the amount of received data at the mobile host is not necessarily equal to the amount of downlink data seen at the monitoring point because of possible losses. For pure TCP traffic we can use the acknowledgement information whether the packets successfully arrived to the mobile, but in a general case it is not possible.

In Fig. 7, a typical available downlink channel rate histogram for GPRS and for the dial-up network is shown.

At the time of the measurements, all networks were configured to use GPRS Coding Scheme-2 (CS-2) [1], which results in ~12 Kbps RLC layer data rate per timeslot. It can be seen that the GPRS system has two dominant channel rate peaks around 21-22 and 34-35 Kbps. This corresponds to the typical cases when the mobile is using two or three GPRS timeslots. We can note that there are only a few four timeslot capable mobiles at the time of the measurement. The results

from the dial-up network indicate more heterogeneous rates around 33 Kbps and between 40-50 Kbps.

The channel rate histogram in a GPRS network can be used to reveal dimensioning related problems. When the peaks around the rate of one or two timeslots (12 and 24 Kbps) are large, it is very likely that the network is overloaded and there is a shortage of free timeslots. The GPRS channel rate histogram in Fig. 7 is from a properly dimensioned network.

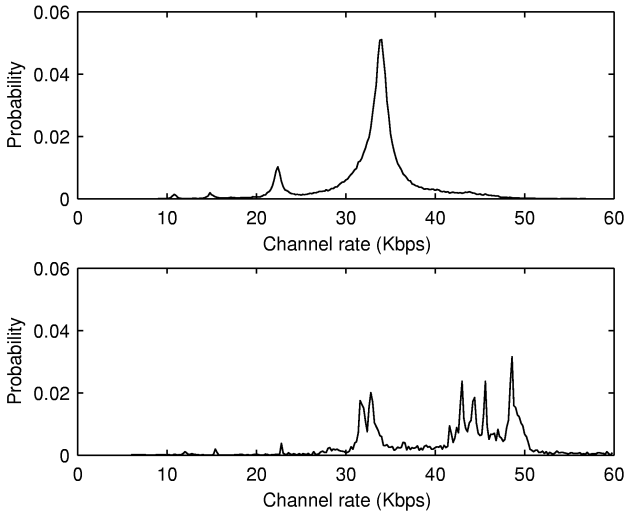

Figure 7. Typical GPRS (top) and dial-up (bottom) available downlink channel rate histograms

V. Packet Loss Estimation

Packet loss estimation algorithms in the literature work either in active [24][25][14] or passive ways [23][26], but all of them rely on active participation of end-hosts. In our case, since we measure in a middle point of the network, neither method is applicable.

In this study, we use an algorithm that works on traces captured in the middle point of the network. The algorithm observes packet sequence numbers and estimates the most likely events and TCP states that might lead to the particular sequence number pattern. More details can be found in [12]. Packet loss is estimated in the downlink direction where most of the data is flowing. The algorithm estimates the loss probability end-to-end and for the two path segments separately: the server to Gi (Internet side loss) and Gi to client (access side loss) paths. This way we can separate the problems in the GPRS network and congestion in the Internet.

Internet side losses happen due to well-known reasons, e.g., buffer overflow or some active queue management algorithms. The estimated loss ratio in the GPRS path segment characterizes the mobile network. It includes possible losses in the Serving GPRS Support Node (SGSN), GGSN, Packet Control Unit (PCU) buffers and RLC/LLC layer (Logical Link Control) losses. Losses may happen due to radio link errors as well as cell reselection (handoff) when some packets stored in the PCU may be lost.

The estimated GPRS side loss (access side loss) is around 1.22% calculated over a long time period and averaged over all networks, see Table II. As expected, the loss ratio in the GPRS access is higher than in the Internet. Under normal conditions, the GPRS access is the bottleneck and this is where packets are lost. The TCP congestion control algorithm tries to fill up the bottleneck buffer in the GPRS network (SGSN, PCU), and because of this, a certain ratio of losses and retransmissions is considered normal.

TABLE II. Packet Loss

	Average GPRS	**Wireline dial-up**
Internet side loss	0.15%	0.39%
Access side loss	1.22%	0.16%

It is possible to increase the buffer sizes and reduce the percentage of losses, but it will also increase the average end-to-end delay. Increasing buffering delay is disadvantageous when several TCP flows share the same channel [2][3]. This trade-off can be analyzed and an optimal setting can be tuned with the assistance of this measurement method.

In addition to average packet loss, it is also important to see how packet losses affect individual TCP connections. One aspect of this is shown in Fig. 8 where the ratio of the different kinds of bottlenecks is depicted. As Fig. 8 shows, 47% of the TCP flows experienced no packet loss at all. Other 48% had more packets lost in GPRS than in the Internet. Only 5% of all TCPs had their bottleneck in the Internet.

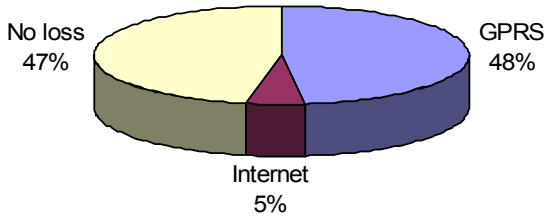

Figure 8. Ratio of TCP flows experiencing no losses, more losses in GPRS than in the Internet (GPRS) and more losses in the Internet than in the GPRS (Internet)

VI. TCP Data Transfer Performance

Measuring TCP data transfer performance is quite straightforward using an active method: one simply has to start a large file download. When using a passive method, we have no control of users' behavior, and we cannot simply calculate the throughput based on any observed TCP connection.

A. Single TCP throughput

In this section, those connections are considered only that did not have any traffic in parallel from the same user during the entire lifetime of the connection. From this set, connections that belong to greedy applications are selected otherwise application layer mechanisms would affect the result and not the actual network capacity. We consider TCP flows belonging to the HTTP protocol holding single, successful request-response pairs.

The connection setup phase was analyzed in the previous section, so we consider the time after the setup has finished, and the first downlink data has been transmitted. It should be also considered that packets might be burst into the bottleneck buffer residing after the monitoring point. Looking at just the downlink data flow very high speeds could be observed, which is not equal to the true end-to-end rate. To compensate this, the transfer time is calculated from the acknowledgements of the first and the last data packets.

The initial slow-start phase is intentionally included in this measure, because it is a significant contributor to the end-user perceived throughput. Nevertheless, to enable TCP to reach the available channel rate, we only consider connections larger than a certain size (100 Kbytes).

The distribution of the calculated single TCP connection throughput is shown in Fig. 9. Observe that the bulk of the distribution is around the channel rate peaks (see Fig. 7). The reasons for non-ideal match are the long slow-start procedure and packet losses.

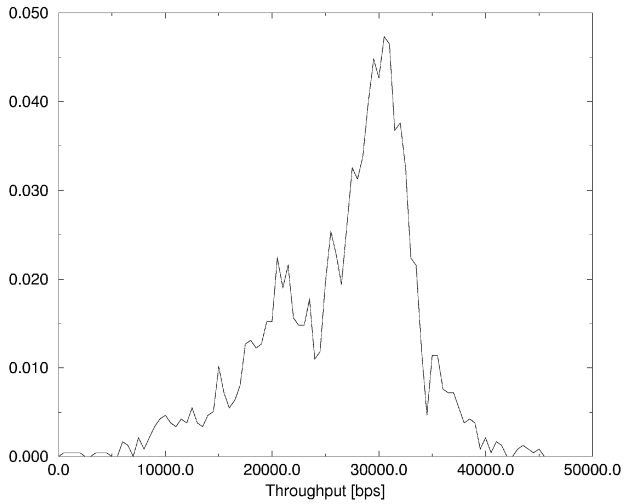

Figure 9. Throughput distribution of single, large TCP flows (>100 Kbytes)

In Fig. 10, we combined the results of the throughput estimation with the loss estimation algorithm. The chart shows the measured average throughput and TCP layer goodput for connections where the ratio of lost packets and sent packets is less than 1%, between 1%-2% and 2%-3% respectively.

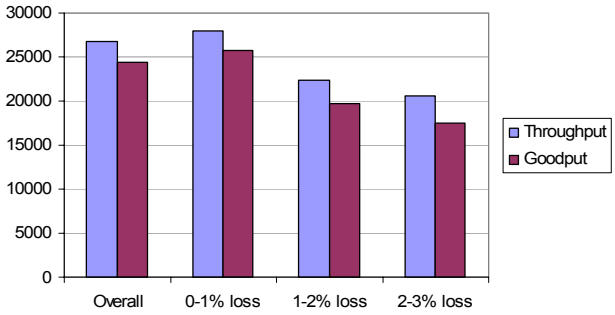

Figure 10. Average throughput and goodput as a function packet loss

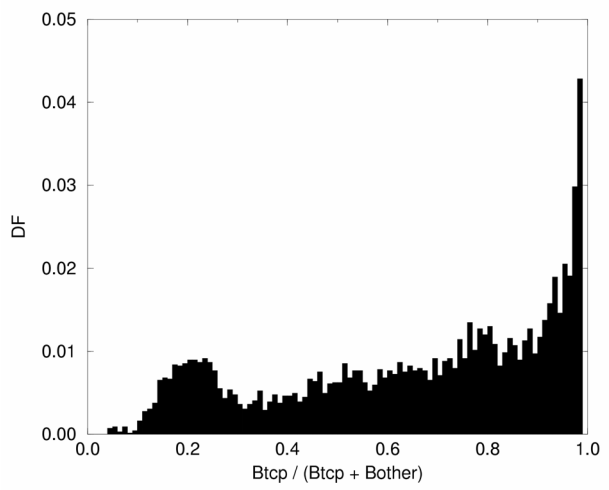

Figure 11. Distribution of the amount of background traffic during TCP flows

B. Throughput estimation in case of parallel background traffic

It is common that the same user has several flows at the same time in parallel to the greedy TCP flow we would like to measure. These flows compete with each other in the same bottleneck buffer and their combined throughput should be considered. We introduce an algorithm to estimate the throughput for this complex case.

Denote the bytes on a long TCP (more than 100 Kbytes) as B_{tcp}. Also denote the total bytes transmitted in parallel to the same user as B_{other}. Fig. 11 shows the distribution of $B_{tcp} / (B_{tcp} + B_{other})$. A ratio around 1 indicates that the TCP was alone, and the ratio towards 0 represents more and more parallel traffic. Observe that most TCPs have considerable amount of parallel traffic, so it is a valid question: "*What is the combined, user perceived throughput?*"

The sequence chart in Fig. 12 shows a situation when packets from the greedy TCP flow (named as tagged and drawn using solid lines) is mixed with packets from other sources to the same mobile user (dashed lines).

We define the following variables for the throughput estimation. T_1 and T_2 denote the time when the first and the last downlink packet holding data are seen at the monitor respectively. Between T_1 and T_2, the sum of bytes from the monitored TCP is counted in B_{tcp}. The bytes from any other sources to the same user are counted in variable B_{other}. The real user throughput is the ratio of the actually received bytes B_m and the associated time $T_m = T_{2m} - T_{1m}$:

$$R = 8\ B_m / T_m.$$

Unfortunately, neither B_m nor T_m can be measured at the monitor, because the monitor is on the other side of the bottleneck buffer.

We assume that the scheduler in the bottleneck buffer maintains the order of packets belonging to the same user (this assumption is valid in all current GPRS systems), so we can approximate

$$B_m = (B_{tcp} + B_{other})\,(1 - P_{loss}) \approx B_{tcp} + B_{other}.$$

The inequality is because some packets might have been lost.

Approximating T_m is not that straightforward. Denote the one-way GPRS transmission delays as D_{up} and D_{down}. Then, T_{1m} and T_{2m} are expressed as:

$$T_{1m} \approx T_1 + D_{down} + T_{buff1}$$

$$T_{2m} \approx T_2 + D_{down} + T_{buff2}.$$

There are extra, non-negligible components denoted by T_{buff1} and T_{buff2}, which account for the buffering times the first and the last packet suffer in the queue. Unfortunately, T_{buff} is difficult to estimate, because it depends on the actual queue occupancy and the unknown bottleneck rate.

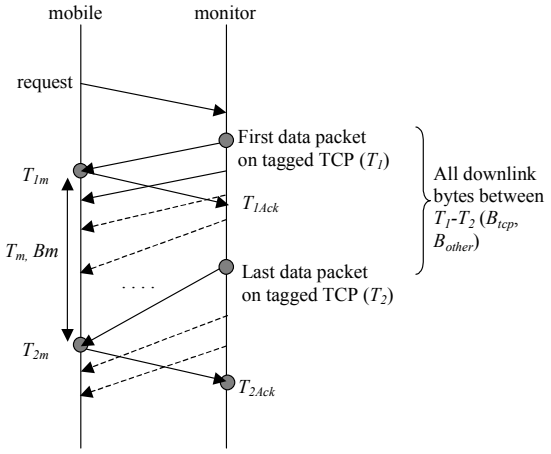

Figure 12. Packet sequence chart for TCP throughput estimation in case there are parallel flows to the same user

So instead, we estimate T_m using the times when the first and the last data packets are acknowledged. Assuming that the uplink channel is not a bottleneck, which is easy to check and it is valid for the vast majority of flows:

$$T_{1m} \approx T_{1Ack} - D_{up}$$

$$T_{2m} \approx T_{2Ack} - D_{up},$$

which leads to:

$$T_m \approx T_{2Ack} - T_{1Ack}.$$

The final approximation for the throughput is:

$$R \approx 8\,(B_{tcp} + B_{other}) / (T_{2Ack} - T_{1Ack}).$$

The nice property of the above formulae is that we can calculate the achieved total throughput to a user as a function of the ratio of the background user traffic $B_{tcp} / (B_{tcp} + B_{other})$.

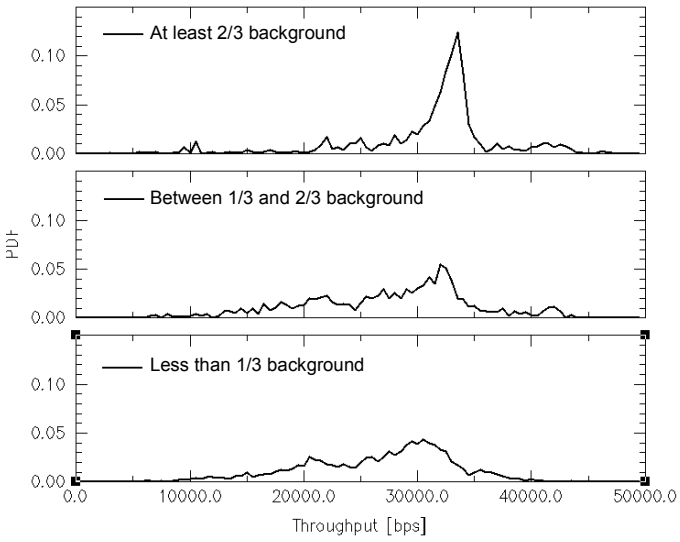

Figure 13. Throughput distribution at three levels of background user traffic

Fig. 13 presents three distributions of the user throughput depending on the amount of background traffic in relation to the greedy TCP flow. The result suggests that the user throughput actually improves and becomes closer to the ideal channel rate as the amount of background traffic increases. This is because several TCP flows in parallel are more aggressive and less sensitive to packet losses and timeouts.

VII. The Effect of TCP Options on TCP Data Transfer Performance

Numerous papers have proposed improvements to the basic TCP algorithm. Due to the less predictable nature of wireless channels, the actual performance of these improvements is of even greater importance. In this section, we are going to quantify the performance improvements of different MSS, window sizes, SACK and TCP timestamp options.

In order to carry out the analysis, we divided bulk TCP downloads into clusters based on MSS, mobile host maximum receiver window size, SACK and TCP timestamp usage. We only considered the performance of single, greedy TCP connections that transferred more than 32 Kbytes of data.

Our aim was to define clusters based on protocol parameters that can be deduced from the traces. These clusters are, however not the only meaningful clusters for the analysis. There can be additional factors that influence TCP performance such as the operating system, initial window size, etc. Moreover, the clustering variables themselves are not statistically independent, so extra care has to be taken when drawing conclusions from the results. Our aim in this section is not to identify the performance aspects of individual TCP parameters, rather to quantify the presence and performance of certain relevant constellations.

The overall results are shown in Table III. In the following subsections, a detailed comparison of clusters is presented.

TABLE III. OVERALL RESULTS FOR DIFFERENT TCP PARAMETERS

	Goodput (bps)	# samples
MSS Low	20048.52	17901
MSS High	24936.86	100654
SACK	25425.25	59750
No SACK	23136.50	64957
Timestamp	25342.24	17739
No timestamp	24049.16	106968
Wnd Low	22542.80	42754
Wnd Med	25723.40	44726
Wnd High	24383.82	37227
Overall	24233.09	124707

A. The effect of MSS

We have calculated the goodput for relevant clusters with low and high MSSs. The results can be seen in Fig. 14. The bars represent the average value, and the ranges represent the standard deviation (not confidence intervals!). The last two bars show the overall goodput with low and high MSS.

It can be seen that having a higher MSS is better for all clusters. The goodput for a 1500-bytes MSS is roughly 15-20% higher than for an MSS of 552 bytes. The reason behind this is twofold. First, the decrease of the header overhead accounts for approximately 6% performance increase. Second, the faster TCP slow start phase and the faster recovery from losses causes the additional performance increase.

The measurements showed that only around 50% of the TCP connections used a large MSS and achieved better performance. However, this ratio will possibly increase in the future, since modern Linux and Windows operating systems set the MSS to 1500 bytes by default.

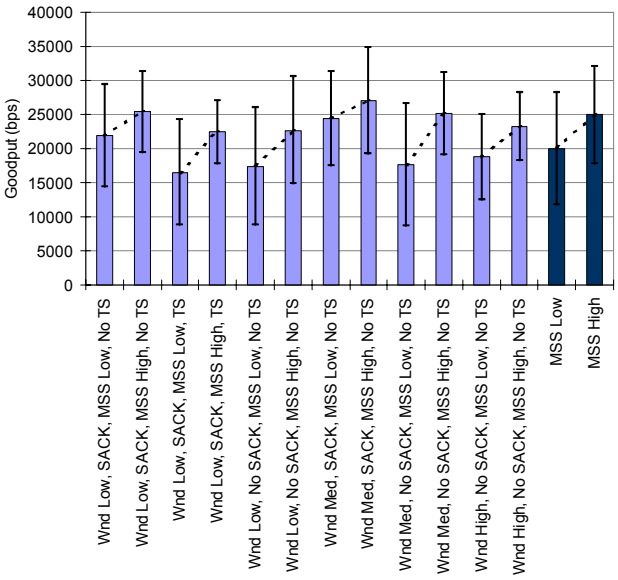

Figure 14. The effect of MSS on TCP data transfer performance

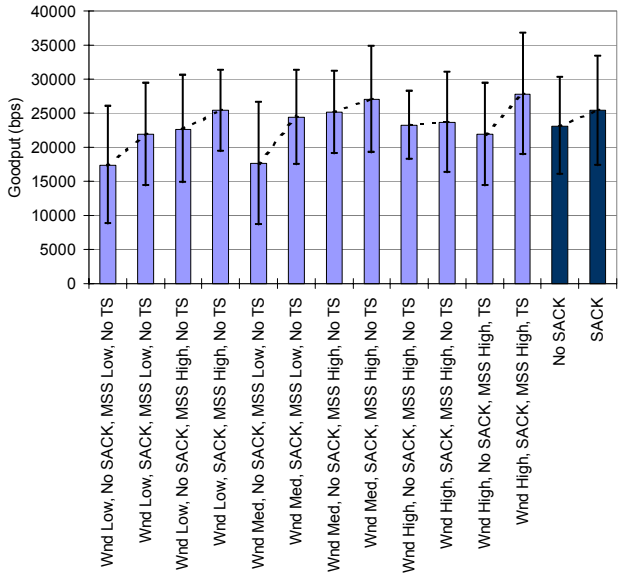

Figure 15. The effect of the SACK option on TCP data transfer performance

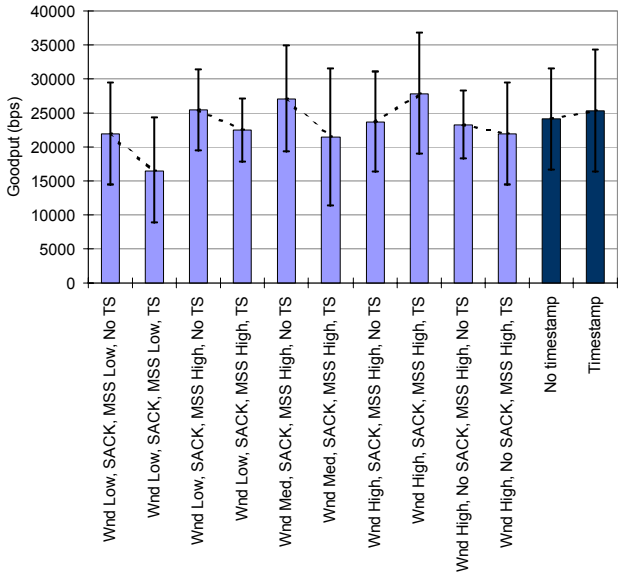

Figure 16. The effect of timestamps on TCP data transfer performance

B. The effect of the TCP SACK option

The mutual effect of a high GPRS RTT and a slow data rate leads to a relatively high TCP RTO value. Consequently, if fast retransmission does not succeed, TCP may need considerable time to fully recover. In this situation, an optimized retransmission strategy, e.g., SACK, may significantly improve the TCP goodput.

In order to prove this, we have performed similar investigations than for MSS, see Fig. 15. On average the TCP SACK option improves the goodput of long TCP transactions by approximately 10%. This improvement was observed for all categories of MSS and window settings.

C. *The effect of the TCP timestamp option*

Fig. 16 shows that overall there is only a minimal difference between timestamped and non-timestamped TCP connections. On the other hand, individual clusters show contradicting results. We suspect that parameters not included in the clustering are more dominant than the timestamp option.

Overall, we were only able to show 2-3% goodput increase. However, in the future, the usage of TCP timestamp option is strongly encouraged because of the advantages of TCP Eifel [5].

D. *The effect of TCP receiver window size*

In [2][3][28] it was concluded that medium sized windows are optimal for GPRS. Small windows are not sufficient because of the relatively large bandwidth delay product. If the window is smaller than 4-5 Kbytes (smaller than the bandwidth delay product) then TCP will be constantly limited by the window and will not reach the available capacity. If the window is less then 9-10 Kbytes (twice the bandwidth delay product) then after a packet loss the pipe may run empty thus reducing the overall TCP data rate.

On the other hand, large windows are not optimal because the bottleneck queue will operate on a higher occupancy level, which has several adverse effects. One such negative effect is that high buffer occupancy increases the probability that TCPs started in parallel time out in the very beginning of the connection since they have not learned the higher RTO value. In addition, a high RTO, in general, harms TCP's ability to recover from losses.

The effect of the maximum receiver window size at the mobile host was investigated by filtering out TCP connections having a window size of: (i) less than 10 KBytes (*Wnd Low*), (ii) between 10 and 20 KBytes (*Wnd Med*) and (iii) more than 20 KBytes (*Wnd High*).

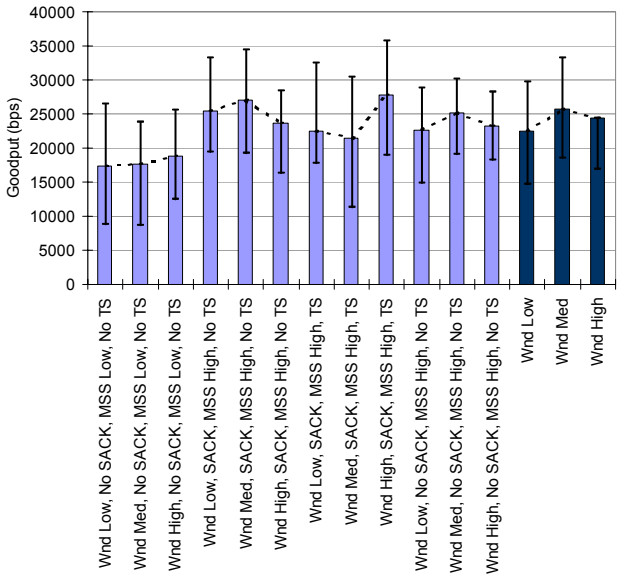

Figure 17. The effect of TCP receiver window size on TCP data transfer performance

The result of the analysis (see Fig. 17) only partly can prove the hypothesis. In some categories (e.g., *SACK, MSS High, TS*) we could even measure opposite results to the recommendation. Nevertheless, the overall result combining all the categories suggests that the medium window size setting does have some 10% improvement considering a large statistical base.

VIII. CONCLUSION

In this paper, we presented a set of passive methods to analyze TCP performance based on non-intrusive traffic monitoring at a middle point of the GPRS network (Gi interface). These algorithms are suitable to analyze TCP connection setup bottlenecks, round-trip delay, GPRS channel rate, TCP throughputs and packet losses. We carried out large-scale statistical measurements in several GPRS networks and compared the results to a wireline dialup network. Using a clustering method, we investigated the impact of certain TCP options and parameters based on a statistically significant dataset. Using an optimized parameter set, one can achieve more than 15% performance gain on average.

REFERENCES

[1] C. Bettstetter, H-J. Vogel, and J. Eberspacher, "GSM Phase 2+; General Packet Radio Service GPRS: Architecture, Protocols and Air Interface", IEEE Communications Surveys, 2(3), Third Quarter 1999.

[2] R. Chakravorty and I. Pratt, "Performance Issues with General Packet Radio Service", Journal of Communications and Networks (JCN) - Special Issue on "Evolving from 3G deployment to 4G definition", pages 266-281, Vol. 4, No. 2, December 2002

[3] R. Chakravorty, J. Cartwright, I. Pratt, "Practical Experience with TCP over GPRS", in Proceedings of the IEEE Global Communications Conference (IEEE GLOBECOM 2002) , Nov. 2002, Taipei, Taiwan.

[4] R. Ludwig and K. Sklower, "The Eifel Retransmission Timer", ACM Computer Communications Review, Vol. 30, No. 3, July 2000.

[5] R. Ludwig and R. H. Katz, "The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions", ACM Computer Communications Review, Vol. 30, No. 1, January 2000

[6] R. Ludwig and A. Gurtov, "Evaluating the Eifel Algorithm for TCP in a GPRS network", In Proceedings of European Wireless, February 2002.

[7] Tom Goff, James Moronski, D.S. Phatak and Vipul Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments", Proceedings of the IEEE INFOCOM 2000.

[8] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgement Options", IETF RFC 2018, October 1996

[9] S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky, "An extension to the selective acknowledgement (SACK) option for TCP", IETF RFC 2883, July 2000

[10] V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance", IETF RFC 1323, May 1992

[11] V. Paxson and M. Allman, "Computing TCP's Retransmission Timer", IETF RFC 2988, November 2000

[12] P. Benko, A. Veres, "A Passive Method for Estimating End-to-End TCP Packet Loss", IEEE Globecom 2002, Taipei, Taiwan

[13] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP", Computer Communication Review, vol. 26, pp. 5-21, July 1996.

[14] V. Paxson, "Automated packet trace analysis of TCP implementations," Proc. ACM SIGCOMM '97, pp. 167-179, September 1997.

[15] J. Mogul, "Observing TCP dynamics in real networks," Proc. ACM SIGCOMM '92, pp. 305-317, August 1992.

[16] V. Paxson, "End-to-end Internet packet dynamics," Proc. ACM SIGCOMM '97, pp. 139-152, September 1997.

[17] P. Barford and M. E. Crovella, "Critical Path Analysis of TCP Transactions," in Proceedings of the 2000 ACM SIGCOMM Conference, Stockholm. Sweden, September 2000.

[18] J. Korhonen, O. Aalto, A. Gurtov, H. Laamanen, “Measured Performance of GSM HSCSD and GPRS”, In Proceedings of the IEEE International Conference on Communications (ICC'01), June 2001.

[19] H. Jiang and C. Dovrolis, “Passive Estimation of TCP Round-Trip Times”, Computer Communications Review, Volume 32, Number 3, July 2002.

[20] M. Meyer, “TCP performance over GPRS”, Proc. IEEE WCNC'99, 1248-1252, 1999

[21] P. Stuckmann, N. Ehlers, B. Wouters, “GPRS Traffic Performance Measurements”, Proceedings of the IEEE Vehicular Technology Conference (VTC 2002 fall), Vancouver, Canada, September 2002

[22] R. Wendland, “How prevalent is Timestamp option in PAWS” e-mail on the end-to-end mailing list, May. 27, 2003.

[23] Y. Tsang, M. Coates, and R. Nowak, “Passive unicast network tomography based on TCP monitoring”, Tech. Rep. TR0005, Rice University, Nov. 2000.

[24] R. Caceres, N.G. Duffield, J. Horowitz, and D. Towsley, “Multicast-based Inference of Network Internal Loss Characteristics”, IEEE Transactions on Information Theory, November 1999

[25] N.G. Duffield, F.L. Presti, V. Paxson, and D. Towsley, “Inferring Link Loss Using Striped Unicast Probes”, IEEE Infocom, April 2001

[26] V. N. Padmanabhan and L. Qiu, “Network Tomography Using Passive End-to-End Measurements”, DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling, Piscataway, NJ, USA, February 2002

[27] R. Kalden, I. Meirick, M. Meyer, “Wireless Internet Access Based on GPRS”, IEEE Personal Communications 7, 8-18, Apr. 2000.

[28] M. Sågfors, R. Ludwig, M. Meyer and J. Peisa, "Queue Management for TCP Traffic over 3G Links", IEEE WCNC 2003, New Orleans, USA, March 2003