Congestion Control in a High Speed Radio Environment

Sara Landström[†], Lars-Åke Larzon[†,‡], Ulf Bodin[†]

{saral, lln, uffe}@csee.ltu.se

[†]Division of Computer Science and Networking
Department of Computer Science and Electrical Engineering
Luleå University of Technology

[‡]Communications Research Group
Department of Information Technology
Uppsala University

***Abstract*—This paper explores interactions between congestion control mechanisms at the transport layer and scheduling algorithms at the physical layer in the High-Speed Down-link Packet Access extension to WCDMA. Two different approaches to congestion control – TCP SACK and TFRC – are studied. We find that TCP SACK and TFRC in most respects perform the same way. SIR scheduling give a higher system throughput for both protocols than RR scheduling, but introduces delay variations that lead to spurious timeouts. The *no feedback timeout* of TFRC was shown to exhibit a similar sensitivity to delay spikes as the retransmit timeout in TCP SACK.**

Index terms: Congestion control, 3.5G, WCDMA.

I. Introduction

The *High-speed Down-link Packet Access (HSDPA) mode*, is part of the 3GPP WCDMA specification release 5 [1]. It supports peak data rates in the order of 10 Mbps with low delays. A key component of HSDPA is the channel scheduler. The channel is divided into 2 ms slots that are assigned to the users according to a scheduling algorithm.

A *round-robin* (RR) scheduler lets users take turns to transmit in an orderly fashion, whereas a *signal-to-interference* (SIR) scheduler gives precedence to the user with the best predicted signaling conditions.

As scheduling is tightly coupled to data availability - which is regulated at a higher level by the transport protocol - we study the interactions between congestion control in the transport layer and channel scheduling in the physical layer.

Of the transport protocols that perform congestion control, TCP is the most widely deployed. Another way of performing congestion control is to apply TCP Friendly Rate Control (TFRC) in which an equation-based model of TCP Reno, derived in [2], is used. TFRC has been designed to give smoother rate changes compared to TCP and is primarily suitable for streaming media applications [3]. An important factor in the send rate equation is the estimate of the round trip time.

In this paper, we study the performance of TFRC and TCP over a HSDPA link layer with both a RR and a SIR scheduler. As expected, SIR is not as fair as RR, but does on the other hand give significantly larger throughput to the users with the best SIRs. We found channel utilization helpful in explaining the observed loads and comparing the two congestion control algorithms. TFRC and TCP performed equally well. Both protocols are however sensitive to delay spikes resulting from SIR scheduling and performance could be improved in this respect.

II. TFRC vs. TCP

TCP SACK (from now on referred to as TCP) is a complete transport protocol with many different features such as congestion control, reliability and session management. In this study, we focus on the mechanisms that affect data availability in lower layers, i.e., the congestion control and avoidance mechanisms.

We compare TCP, to the alternative approach to congestion control given by TFRC. There are a number of fundamental differences between TCP and TFRC - TCP is sender-oriented and uses a

sliding window to control the send rate whereas TFRC is receiver-oriented and uses an equation-based scheme. In the following sections we briefly describe the transport layer mechanisms that we study in this paper, and how they differ between TFRC and TCP.

A. Adaptive timeouts

If the sender does not receive an acknowledgment of transmitted data before a timeout, the sending rate is reduced as timeouts are interpreted as signs of network congestion. Both TFRC and TCP use a moving average filter on RTT samples to calculate an RTT estimate that controls the timeout values. However, where TCP reduces the sending rate to a minimum after a timeout and forces the sender into *slow start*, TFRC only halves the sending rate.

In TFRC, the timeout is called a *no feedback timeout*, and it is only a supplement to the retransmit timeout of TCP, which is modeled by the send rate equation that TFRC complies to. Therefore the no feedback timeout can have a longer interval (at least four times the estimated RTT), whereas the retransmit timeout is the estimated RTT with a margin accounting for RTT fluctuations.

The Eifel algorithm, presented in [4], has the potential to improve performance in wireless systems by making it possible to faster regain the send rate used before a timeout if the acknowledgment causing the timeout arrives late. Timeouts that would not have occurred if the timeout interval had been longer are called spurious. The actions to take after a spurious timeout is however still being debated [5].

B. Congestion avoidance

During normal operation, i.e, when the sender has left the slow start phase, the send rate is increased slowly to avoid filling up queues too fast. In TFRC, the receiver detects lost packets through the arrival of three later sent packets. Packet losses during one RTT only result in one congestion event, as in TCP. The TFRC receiver informs the sender of the perceived loss rate, p, and the receive rate, X_{recv}, which the sender uses to calculate a tentative new sending rate, X_{calc} using equation 1.

$$X_{calc} = \frac{s}{R + f(p)} \quad , \tag{1}$$

where s is the mean packet, R the round trip time and $f(p)$ is given by

$$f(p) = \sqrt{2 * \frac{p}{3}} + 12 * \sqrt{3 * \frac{p}{8}} * p * (1 + 32 * p^2).$$

To determine the new sending rate, the computed sending rate is compared to the receive rate according to equation 2

$$X_{send} = min(X_{calc}, 2 * X_{recv}). \tag{2}$$

As TCP uses cumulative acknowledgments, a duplicated acknowledgment (dupack) indicates an out-of-order reception of data. Upon reception of four successive identical acknowledgments, i.e., three dupacks, TCP assumes a lost packet and halves its sending rate.

C. Slow start

In the beginning of each transfer and after a timeout in TCP, the session is in a *slow start* phase. During this phase, the send rate increases exponentially until a lost packet is detected. In TCP, this is accomplished by increasing the window that controls the send rate proportionally to the number of acknowledgments received.

TFRC mimics the TCP slow start behavior in the beginning of each session by doubling the sending rate once per round trip time, if the reported receive rate matches the current send rate, otherwise the new sending rate is set to twice the receive rate. This means that the sending rate is limited only by the current values of X_{send} and X_{recv} until the first loss event.

III. Evaluation

Our evaluation of TFRC and TCP over the high speed down-link channel (HS-DSCH) in HSDPA is based on simulations. Performance is investigated both for an RR and a SIR scheduler using two different loads. This results in four scenarios for each congestion control mechanism.

A. Model

The network simulator (ns) version 2.27 was complemented with module for simulating HSDPA [6]. We also modified the TFRC code to follow RFC3448 and measures were taken to remove the bug reported in [7].

TABLE I
SUMMARY OF RADIO MODELS AND PARAMETERS

Phenomena	**Model/Configuration**
Path loss	Exponential, propagation constant 3.5
Shadow fading	Stddev 8dB
Self interference	Limited to 10%
Intra cell interference	Limited to 40%
Inter cell interference	Limited by distance
ARQ	No, immediately retransmitted
Code multiplexing	Max 3 users
BLER	Uniformly distributed 10% for SIR over -3.5dB, lower SIR 50%

Table I gives an overview of the radio models implemented and their configuration.

There is no fast power control over the high speed shared channel, instead link adaptation is employed. The combination of coding rates and modulation types included in the simulator are introduced in Table II. Note, with these combinations a maximum bit rate of 7.20 Mbps can be achieved. We assume that the number of spreading codes and the power assigned to HS-DSCH, change on long time scales compared to the simulation time. The average power was fixed to 10 W and 12 out of 16 codes were used.

TABLE II
COMBINATIONS OF CODING RATES AND MODULATION TYPES

Coding (rate)	**Modulation** (type)	**SIR** (dB)	**Bitrate** (Mbps)	**Radio block** (bytes)
0.25	QPSK	-3.5	1.44	360
0.50	QPSK	0	2.88	720
0.38	16QAM	3.5	4.32	1080
0.63	16QAM	7.5	7.20	1800

Seven cells with omni-directional antennas and a 500 m radius were simulated and the performance in the center cell was analyzed. A fixed delay was used to model the delay over the wired links between the sources and base stations. The delay, 75 ms, was the same in both directions. The wired links were over provisioned such that the bottleneck was over the wireless link. When reaching the base station user data was stored in individual buffers, each capable of keeping 90 IP packets. This means that it is possible for a single user to capture the wireless channel with SIR scheduling. With our simulation set-up, packets can only be lost in the queue awaiting transport over the wireless link.

We varied the load by simulating either 50 or 65 (30% more) stationary mobile terminals present in the coverage area. The nodes were distributed uniformly over the seven cells. The effects from scheduling showing at this load, would probably become apparent at higher loads with better tuned scheduling algorithms than RR and SIR. Alternatively, the load could have been varied by using different average waiting times.

Every session consisted of a mobile down loading a file followed by a truncated exponentially distributed waiting time with mean 2 seconds and a minimum value of 0.5 seconds. The waiting time was initiated as soon as all the data had reached the receiver[1]. The file sizes were randomly chosen from nine possible sizes where the number of segments i is given by equation 3.

$$i_n = 2i_{n-1} + 1,\ n = 1, 2...9,\ i_0 = 0 \qquad (3)$$

A fixed payload size of 1450 bytes was used, in order to create the same number of packets for both TCP and TFRC. The relation between the frequencies with which the file sizes were likely to be selected was 1:2:3:4:5:6:7:8:9, where 9 corresponds to the smallest file size. Two relatively large file sizes have been included, i.e., 740950 and 1483350 bytes. The reason was threefold, first for the short file sizes slow start is essentially never left. Secondly, the schedulers' behaviors have larger impact on longer file transfers and finally TFRC is targeted at longer lived sessions.

TFRC does not include connection establishment nor tear-down, which for short flows result in higher throughput. Therefore, to enable comparisons, TCP was configured to send data with the initial SYN segment. By setting the initial window of TCP to two segments and sending data on the SYN, the window is doubled as if no handshake was made. A minimum retransmit timeout of 1 second and a timer granularity of 0.01 seconds were used for TCP.

[1] We reset the transport layer endpoints after receiving the last piece of data.

Five minutes system time was simulated in each run and all scenarios were repeated twenty times. The random number generators giving the positions of the mobiles, the starting times of the transfers and the file sizes were given different seeds in each simulation of the same scenario.

IV. Results

When analyzing the material, we aggregated data from all the simulations of the same scenario. The data from the first 5 simulated seconds in each run was removed to avoid initialization bias. Only performance in the center cell was studied. In most cases the results obtained for both loads were similar, hence if nothing else is stated the figures represent the case when there are 65 mobiles present in the system. The confidence intervals are for 90%.

We have used two approaches when performing the analysis. First, we look at transport layer events, thereafter we study when data is available to the scheduler.

A. Transport layer events

Packet loss – To TFRC, which offers unreliable transport, the packet loss rate is an important metric. If the loss rate is too high the data which reaches the receiver is not going to be useful. Bursty losses may also be problematic depending on the application. However, when we study the share of packets that was lost for TFRC the purpose is to detect connections between the transfer sizes and their loss rates that help our analysis. For instance, the transitions between slow start and congestion avoidance are controlled by the packet loss events and how they are detected. An inspection of the packet loss rates experienced by TFRC reveals that there are basically three groups of file sizes, which we will call small, medium and large.

Small: In the first group, the six smallest transfer sizes fall. These flows do not loose any packets and hence never leave slow start due to packet loss.

Medium: The 369750 bytes flows form a group of their own. Approximately 80% of these transfers get through intact. The remaining 20% of the flows loose less than a tenth of the data.

Large: For the two largest file sizes the transition from slow start to congestion avoidance can not be avoided. In the case of 740950 bytes, the flows all reach the maximum available capacity and about 15% have high loss rates (20%), since the file sizes are still rather short. For the 1483350 byte flows the loss rate stays below 10%, except in the case of SIR scheduling and 65 mobiles, where 2% of the flows experience higher loss rates. These flows are long enough to compensate for the high loss rates encountered when probing for an initial estimate of the capacity.

We expect to find that the TCP flows exhibit the same behavior, since the slow start strategies are similar.

Detection of packet losses – Long propagation delays potentially result in each source having large amounts of data residing in the network. Only parts of this data at a given time is actually in the bottleneck buffer. If packets travel closely together buffer overflow will occur at lower send rates. TCP is window-based and can send several segments back to back. TFRC on the other hand, space out the packets over a round trip time, thereby decreasing the probability that a large part of the data in flight finds itself in the queue at the edge of the wireless link simultaneously.

With TCP, the window has to be larger than the bottleneck buffer size to sustain any losses, since new data must be acknowledged before a new segment is released. It is most vulnerable to timeouts when quickly increasing the send rate during slow start, since many segments can then be lost from the same window, such that three duplicated acknowledgments are not generated as is needed for a fast retransmission to be made. When a timeout occurs, a parameter called the slow start threshold, is set to half the current window, forcing TCP into congestion avoidance whenever the window size exceeds this value. Since TCP increases its rate slower during congestion avoidance and the flows are buffered individually, later timeouts are connected to decreased bottleneck capacity - leading to an assembly of segments in the sensitive buffer.

In Figure 1 the average number of timeouts per transfer size with TCP is shown. The larger number of timeouts with SIR scheduling is caused by changes in the available capacity due to competing sources and not the probing behavior of TCP. Of the total number of timeouts with TCP and SIR scheduling close to 70% were spurious.

Due to the fact that it is enough for one packet to reach the receiver to allow the next acknowledg-

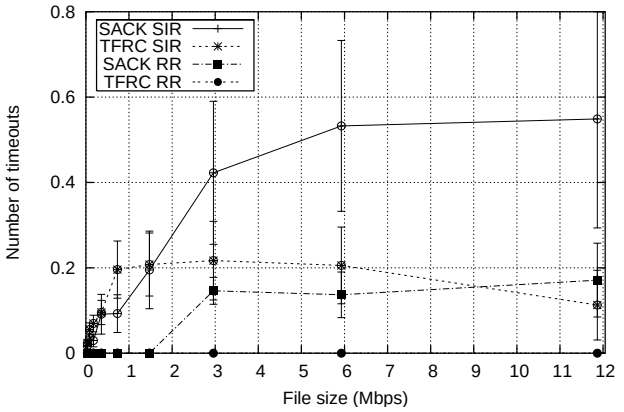

Fig. 1. Timeout events for TCP and TFRC.

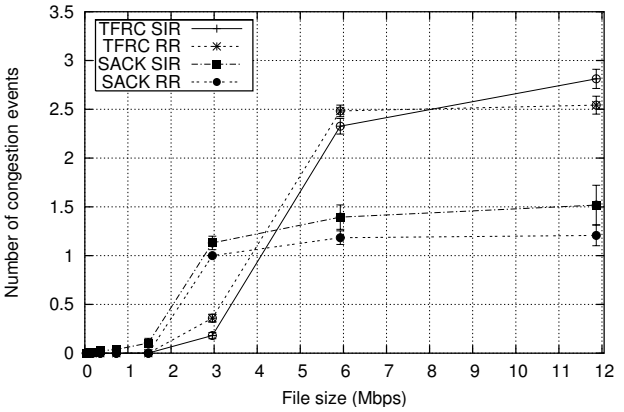

Fig. 2. The average number of congestion events detected through later sent packets arriving at the receiver in TFRC, or through three duplicate acknowledgments in TCP.

ment to be sent in TFRC and the relatively large interval, *no feedback timeouts* are rare, see Figure 1. No timeouts of this type were observed for TFRC with RR. With SIR scheduling there were a few occurrences, but they were considerably fewer than the TCP retransmit timeouts. The no feedback timeouts can not be said to be spurious, since they prevent data from being sent continuously at the same rate when it is not getting through. The primary congestion detection mechanism in TFRC is however the correspondence to TCP's three dupack mechanism.

Since there is no packet reordering possible with our network configuration, the arrival of three duplicated acknowledgments in TCP confirms the loss of at least one segment[2]. TCP then has a chance to retransmit the presumably lost segment, before the timeout expires. For TFRC, we observe few congestion events being detected by later sent packets arriving in relation to the loss rates. Hence, we conclude that the losses are often correlated and occur in bursts during a round trip time. This type of events are more frequent for TFRC for the two largest file sizes than the fast retransmits are for TCP, which can be explained by TFRC being slower to reduce its send rate. The situation is reversed for the 740950 byte files, which is likely a consequence of TCP sending its segments back to back instead of spacing them out as TFRC does. Thus, TCP encounters its first losses earlier. The three dupack events and the packet losses that lead to congestion events in TFRC are summarized in Figure 2. The confidence intervals for this event are narrow and there is no apparent difference between the schedulers.

[2] Retransmissions can trigger a large number of duplicate acknowledgments, such that a fast retransmit is performed when not needed.

B. Data availability

For system throughput, having data to be transferred at the highest data rate in every slot represents the optimal situation. However, only the SIR scheduler can take advantage of having several potential receivers with different SIR conditions to choose between. A requirement for good throughput, which is independent of scheduler characteristics, is that there should be at least one potential receiver in each slot. Hence, we focus on the distribution of the number of potential receivers for this study.

The distribution of the number of potential receivers for TCP is shown in Figure 3 and for TFRC in Figure 4. As can be seen, there are few slots that are not used. SIR scheduling gives a larger concentration around 1-2 potential receivers than RR scheduling does and the right tail is thinner for SIR than for RR.

In general the SIR scheduler gets more files through than the RR scheduler with the same number of users in the system. The current application model, where the waiting period is initiated as soon as the transfer has been completed, leads to high SIR users generating a larger part of the data with

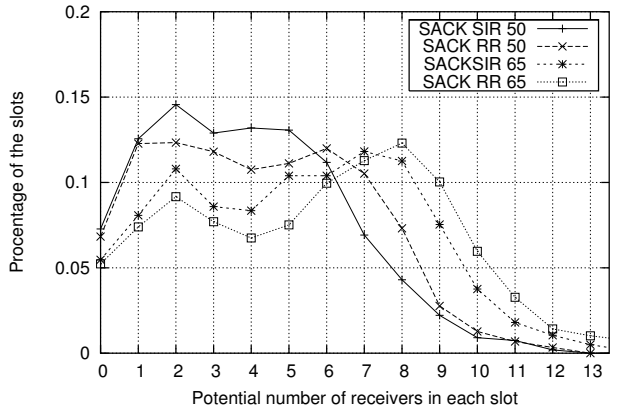

Fig. 3. Distribution of the number of potential receivers for TCP.

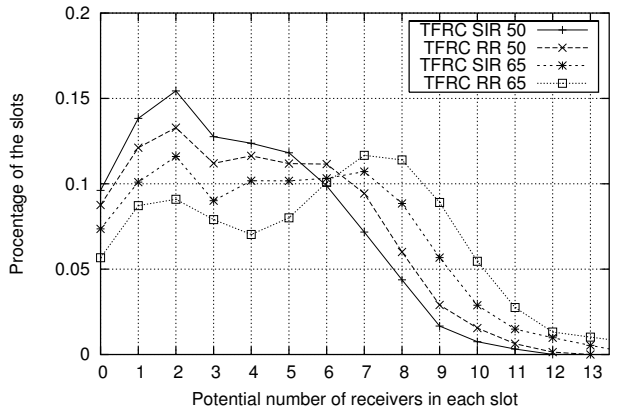

Fig. 4. Distribution of the number of potential receivers for TFRC.

SIR scheduling than with RR scheduling. This is not an unlikely scenario however, since poor SIR users might wait for an indication of improved signal reception before attempting transfers. The application model does however discriminate against the RR scheduler and we would like to try other application models in the future.

When comparing the number of bytes transferred with TCP and SIR scheduling for 50 mobiles and RR for 65 mobiles, the difference in result is smaller than when comparing the system throughput with the same number of mobiles for the two schedulers. This indicates that a RR scheduler needs a larger number of mobiles to generate the same offered load, i.e., number of transfers. With TFRC the RR scheduler never reaches the same levels as the SIR scheduler. Since TFRC is

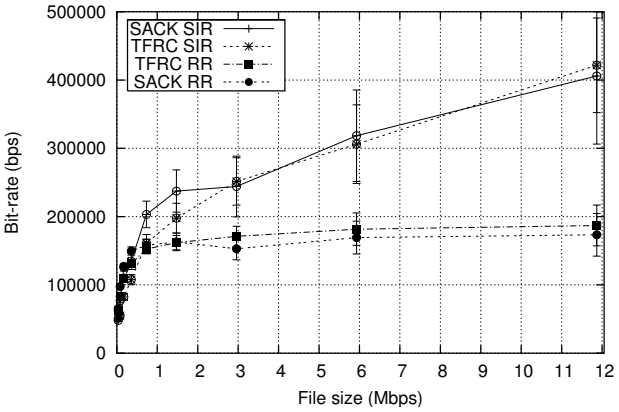

Fig. 5. Average transfer rate for TCP and TFRC.

unreliable, the artifacts that might come of SIR scheduling, i.e., higher loss rates, does not lead to retransmissions and as severe send rate reductions. Therefore it is likely that a higher offered load can be sustained and that the limiting factor may be the influence of the loss rate on the quality.

C. Transfer rates

The most important metric for a file transfer is the obtained transfer rate. The transfer rates in Figure 5, puts the previously observed events into perspective. The difference between the two protocols is small, although the difference in performance between the schedulers is big.

A metric used in [8] for determining whether the system had appropriate settings is the 5th percentile transfer rates. According to this metric, the HSDPA channel is supposed to do at least as well as the Circuit Switched Equivalent for web browsing, which means that 95% of the users should have a bit rate exceeding 50 kbps. We found that this condition was met for both protocols. When looking at the 5th percentile bit rates on a per flow size basis, Figure 6 and Figure 7, we find that it is the small flows that do not reach 50 kbps. RR scheduling gives higher transfer rates to this group of flows than SIR, but the RR scheduler operates at a lower offered load with the current application model.

V. Discussion

Future studies include investigating a range of propagation delays for the wired links in the path and different buffer strategies at the wireless chan-

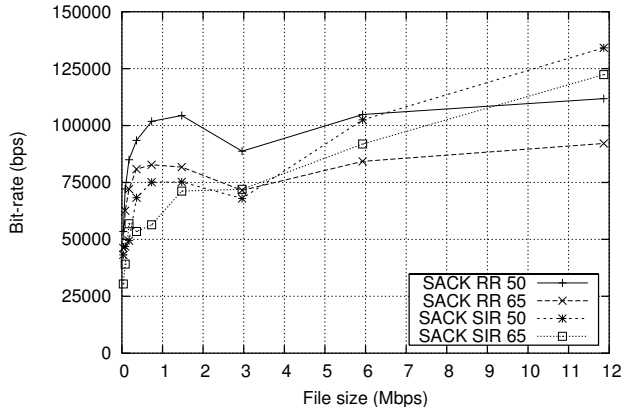

Fig. 6. 5th percentile transfer rates for TCP.

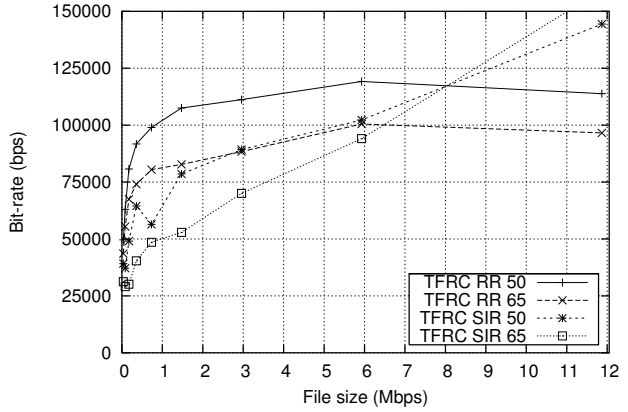

Fig. 7. 5th percentile transfer rates for TFRC.

nel. With these additional dimensions in the evaluation follows needs to more accurately track delay spikes and their effect on the probability for packet loss. Finding appropriate buffer sizes, that balance the risk of buffer overflow and long queuing delays for wireless channels and different applications is non-trivial. Especially if TFRC and TCP are to co-exist in an environment where the available channel capacity can vary substantially. In this study we used the same application model for both TFRC and TCP, in the future we would like to include a model of a streaming application for TFRC and look at other ways to distribute the transfers among the mobiles.

VI. Conclusions

We have performed an initial investigation of how congestion control at the transport layer, lead to different physical channel utilization patterns for a high-speed shared wireless cellular environment. We have found, that with an application model where the waiting time is initiated as soon as a transfer is finished, the observed load is both the result of the nature of the scheduling algorithms for the shared environment as well as the congestion control algorithms.

As expected, the SIR scheduler gives higher average transfer rates at the expense of fairness compared to the RR scheduler. The main reason being that the channel is better utilized since the average transfer times are shorter, which leads to a faster initialization of the following transfers. As high SIR users complete their transfers faster with the SIR scheduler, a larger part of the generated load comes from these users. In general, a higher load is created for the same number of mobiles with SIR scheduling than with RR scheduling.

The difference in transfer rates between TFRC and TCP is small, although the system throughput is higher with TFRC. This can be explained by the distributions of the number of the potential receivers being similar, thus the retransmissions performed by TCP take up capacity corresponding to the additional transfers performed with TFRC.

We conclude that the common type of application model used in this study lead to offered loads that depend on algorithms both at the transport and the physical layer. It is however not unreasonable, since users are likely to transfer more data if they get fast responses.

References

[1] Hämeenlinna. Overview of 3GPP Release 5. Technical Report 030375, ETSI Mobile Competence Center, Jun. 2003.

[2] J. Padhye, V. Firiou, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *ACM Sigcomm*, 1998.

[3] Sally Floyd, Mark Handley, Jitendra Padhye, and Jörg Widmer. Equation-Based Congestion Control for Unicast Applications. In *ACM Sigcomm*, 2000.

[4] R. Ludwig and R. H. Katz. The Eifel Algorithm: Making TCP Robust Against Spurious Retransmissions. *ACM Computer Communications Review*, 30(1), jan 2000.

[5] Andrei Gurtov and Reiner Ludwig. Responding to Spurious Timeouts in TCP. In *IEEE INFOCOM*, 2003.

[6] Ulf Bodin and Arne Simonsson. Effects on TCP from Radio-Block Scheduling in WCDMA High Speed Downlink Shared Channels. In *QoFIS*, 2003.

[7] Sally Floyd. A bug in the TFRC code in NS, 2003.

[8] Janne Peisa and Eva Englund. TCP performance over HS-DSCH. In *IEEE VTC*, May 2002.