Influence of Power Control and Link-Level Retransmissions on Wireless TCP

Niels Möller and Karl Henrik Johansson

KTH, Stockholm*

Abstract. A fundamental assumption of the TCP protocol is that packet losses indicate congestion on the network. This is a problem when using TCP over wireless links, because a noisy radio transmission may erroneously indicate congestion and thereby reduce the TCP sending rate.

Two partial solutions, that improve the quality of the radio link, are power control and link-level retransmissions. By modeling these two lower layers of control loops, we derive an analytical model of the delay distribution for IP packets traversing a link. We investigate the effect on TCP, in particular the performance degradation due to spurious timeouts and spurious fast retransmits caused by delays and reorder on the link. It is shown that the models allow us to quantify the throughput degradation. The results indicate that link-level control and TCP interact, and that tuning one or the other is needed in order to improve performance.

1 Introduction

The TCP algorithms assume that packet losses indicate congestion on the network [1,2]. When using TCP over wireless links, packet drops due to radio noise makes TCP reduce its sending rate, hurting performance [3,4,5,6]. Two partial solutions, that improve the quality of the radio link, are power control and link-level retransmissions.

Power control tries to adapt the transmission power to varying channel characteristics, "fading", which can be thought of as a disturbance. Link-level retransmission, such as Automatic Repeat Request (ARQ), tries to hide losses from upper layers, e.g. TCP/IP, by resending damaged radio blocks. The IP packets still, however, experience random delays or even reorderings when they are transmitted across such a link. How to deal with these problems on the TCP level is the topic of intensive research, e.g., [6,7,8,9]. Despite recent progress, the influence of the radio link properties on TCP is far from fully understood.

In this article, we use models for these two link-layer processes, and derive the link properties, in particular the delay distribution, experienced by IP-packets. From these properties, we derive the effects on TCP throughput.

* This work was supported by the Swedish Research Council and the Swedish Center for Internet Technologies through the SRI project.

G. Karlsson and M.I. Smirnov (Eds.): QofIS 2003, LNCS 2811, pp. 193–202, 2003.

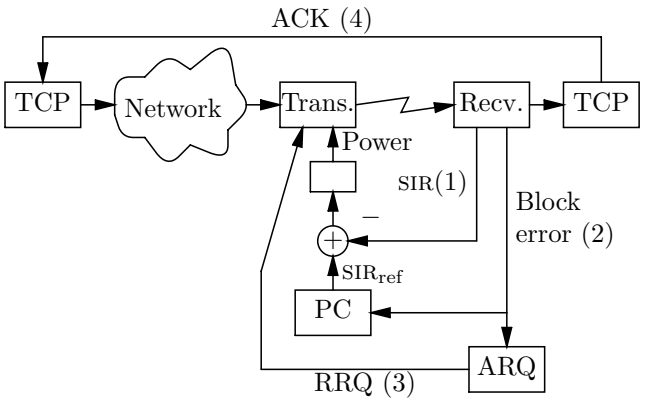

Fig. 1. System overview, including four feedback loops marked (1) – (4).

1.1 System Overview

When using TCP over a wireless link, there are several interacting control systems stacked on top of each other, illustrated in Figure 1. At the lowest level, the transmission power is controlled in order to keep the signal to interference ratio (SIR) at a desired level. This is a fast inner loop intended to reject disturbances in the form of "fading", or varying radio conditions. On top of this, we have an outer power control loop that tries to keep the block error rate (BLER) constant, by adjusting the target SIR of the inner loop. The radio channel and power control are modeled in Section 2. Next, we have local, link-level, retransmissions of damaged blocks, described in Section 3. Finally, we have the end-to-end congestion control of TCP. The effects the link layer control has on TCP performance is investigated in Section 4.

2 Radio Model

Data is transmitted over the radio link as a sequence of *radio blocks* (RB). One radio block corresponds to a *transmission time interval* (TTI) of 10 or 20 ms. Depending on the bandwidth (typically from 64 kbit/s to 384 kbit/s) the size of a RB can vary from 160 octets (the small 10 ms TTI is not used for the lowest data rates) to 960 octets.

The transmission of the radio blocks is lossy. Let p denote the overall probability that a radio block is damaged. The power of the radio transmitter is controlled, so that the loss probability stays fairly constant. The target block error rate is a deployment trade-off between channel quality and the number of required base stations. For UMTS the reference block error rate is often chosen to be about 10%, see [10]. In the following, we thus assume $p = 0.1$.

2.1 Power Control

The typical power control uses an inner loop that tries to keep the signal to interference ratio (SIR) close to a reference value SIR_{ref}. This loop typically has a sample frequency of 1500Hz, and a one bit feedback that is subject to a delay of two samples, i.e. 1.3 ms. In simulations [11], the inner loop is able to track the reference SIR within 2-3 dB, with a residual oscillation due to the severe quantization. The period of this oscillation is typically less than 5 samples, i.e. 3.3 ms.

As there is no simple and universal relationship between the SIR and the quality of the radio connection, there is also an outer loop that adjusts SIR_{ref}. This loop uses feedback from the decoding process; in this article we assume that the power control outer loop is based on block errors.

As it is hard to estimate the BLER accurately, in particular if the desired error rate is small, one approach is to increase SIR_{ref} significantly when an error is detected, and decrease the SIR_{ref} slightly for each block that is received successfully. It is interesting to note that this strategy resembles the TCP "additive increase, multiplicative decrease" congestion control strategy. For a survey of modern power control techniques for systems such as WCDMA, see [11].

2.2 Markov Model

The outer loop of the power control sets the reference value for the SIR. Given a particular reference value $\text{SIR}_{\text{ref}} = r$, the obtained SIR is a stochastic process. Together with the coding scheme for the channel, we get an expected probability for block errors. If the coding scheme is fixed, the probability of block errors is given by a function $f(r)$.

One can compute $f(r)$ from models of the channel and coding. For the BPSK example given in [12] we get

$$f(r) \approx 1 - \left(1 - Q(\sqrt{2\alpha e^{\beta r + \frac{1}{2}\beta^2\sigma^2}})\right)^N \tag{1}$$

where α is the coding gain, $\beta = \frac{\ln 10}{10}$, σ is the standard deviation of the received SIR, N is the number of bits in a radio block, and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-x^2} \mathrm{d}x$. We take $\alpha = 4$, from [12], $\sigma = 1$ dB, corresponding to a small SIR oscillation as in [11], and $N = 2560$, corresponding to a modest link capacity of 128 kbit/s.

However, for control purposes, even a crude approximation of the true $f(r)$ should be sufficient. In general, $f(r)$ is a decreasing, threshold-shaped function. For small enough r, $f(r) \approx 1$, i.e. almost all blocks are lost. For large enough r, $f(r) \approx 0$, i.e. almost no blocks are lost. The operating point of the power control is the point close to the end of the threshold where $f(r) = p = 0.1$, so it is the shape of $f(r)$ close to this point that is important.

The outer loop of the power control uses discrete values for SIR_{ref}. One way to keep the block error probability close to the desired probability p, is to use a fixed step size Δ as follows. Whenever a radio block is received successfully, SIR_{ref}

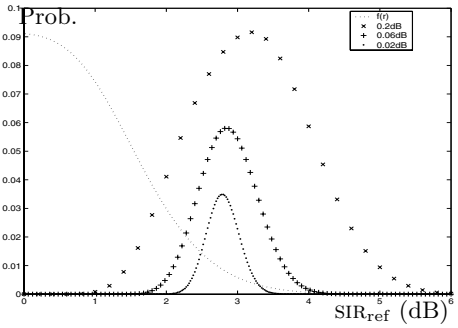

Fig. 2. Stationary distribution for the power control. Each mark represents one state of the power control, the corresponding value of SIR_{ref}, and its stationary probability. The dotted curve is the threshold-shaped function $f(r)$, scaled to fit in the figure, which represents the block error probability as a function of SIR_{ref}.

for the next block is decreased by Δ. And whenever a radio block is damaged, SIR_{ref} for the next block is increased by $K\Delta$, where $1/(1+K) = p$. The value of Δ is an important control parameter, which determines the performance of the power control.

For an integer K the varying SIR_{ref} can be viewed as a discrete Markov chain [12]. To do this, we have to make the assumption that block errors depend only on the value of SIR_{ref} when the block is transmitted; there is no relevant history except the SIR_{ref}. In our case $p = 0.1$ implies $K = 9$.

For a given function $f(r)$ and a step size Δ, we also modify f by setting $f(r) = 1, r < r_{\min}$ and $f(r) = 0, r > r_{\max}$. This results in a finite Markov chain, and it is straight forward to compute the stationary distribution for SIR_{ref}.

Figure 2 shows the threshold function $f(r)$, together with the stationary distribution for three values of Δ. Note that a smaller step size gives a more narrow distribution, and a smaller average SIR, which means that there is a trade-off between energy efficiency and short response times.

For a large Δ, the power control state, i.e. SIR_{ref}, will move quite far along the tail of the $f(r)$ threshold. For a smaller Δ, the control state will stay close to the operating point, so that a linearization of $f(r)$ would be a good approximation.

3 Link-Layer Retransmissions

The simplest way to transmit IP packets over the wireless link is to split each IP packet into the appropriate number of radio blocks, and drop any IP packet where any of the corresponding radio blocks were damaged. But as is well known, TCP interprets all packet drops as network congestion, and its performance is therefore very sensitive to non-congestion packet drops. An IP packet loss probability on the order of 10% would be detrimental.

There are several approaches to recover reasonable TCP performance over wireless links. In this paper we concentrate on a local and practical mechanism:

The link can detect block damage (this is needed for power control anyway), and it can use that information to request that damaged blocks be retransmitted over the link. This capability is an option in standard wireless network protocols, see [13] for an evaluation of these options in the IS-2000 and IS-707 RLP standards. Alternative approaches include changes to the TCP algorithms (e.g. Eifel, [7], and TCP Westwood [6]), and the use of forward error correction coding to add redundancy to the packets, either end-to-end as in [14] or at the link as in [15].

The effect of link level retransmissions is to transform a link with constant delay and random losses into a link with random delay and almost no losses.

There are several schemes for link level retransmission. We will consider one of the simpler, the (1,1,1,1,1)-Negative Acknowledgement scheme [16], which means that we have five "rounds", and in each round we send a single retransmission request. When the receiver detects that the radio block in time slot k is damaged, it sends a retransmission request to the sender. The block will be scheduled for retransmission in slot $k + 3$ (where the delay 3 is called the RLP NAK guard time). If also the retransmission results in a damaged block, a new retransmission request is sent and the block is scheduled for retransmission in slot $k + 6$. This goes on for a maximum of five retransmissions.

Consider the system at a randomly chosen start time, with the state of the power control distributed according to the stationary distribution. For any finite loss/success sequence (for example, the first block damaged, the next six received successfully, the eighth damaged), we can calculate the probability by conditioning on the initial power control state and following the corresponding transitions of the Markov chain. In the following sections, we use these probabilities to investigate the experience of IP packets traversing the link.

4 TCP/IP Properties

When transmitting variable size IP packets over the link, each packet is first divided into fix size radio blocks. We let n denote the number of radio blocks needed for the packet size of interest. For the links we consider, we have $n \leq 10$.

The outermost feedback loop we consider is the end-to-end congestion control of TCP. The role of TCP is to adapt the sending rate to the available bandwidth. The inputs to the TCP algorithms are measured roundtrip time, and experienced loss events, both of which depend on the links traversed by the TCP packets. To understand and predict the behavior of TCP, we must consider how it interacts with the power control and the link level retransmissions.

4.1 IP Packet Delay

One important link property for TCP is the delay distribution of packets traversing the link. Using the loss/success sequence probabilities from Section 3, we can explicitly compute the IP packet delay distribution. The expected value and standard deviation for three values of the step size are shown in Table 1. The times are measured in TTIs, and only the delay due to retransmissions is included (an n-block packet is of course also subject to a fix delay of n TTI).

Table 1. Mean and standard deviation of the IP packet delay distribution.

n	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
1	0.31±0.94	0.32±0.99	0.33±1.03
2	0.51±1.08	0.53±1.15	0.54±1.20
3	0.62±1.08	0.64±1.18	0.65±1.24
4	0.72±1.09	0.74±1.21	0.76±1.28
5	0.83±1.09	0.85±1.23	0.87±1.31
6	0.94±1.09	0.96±1.25	0.98±1.35
7	1.06±1.09	1.07±1.27	1.09±1.38
8	1.17±1.09	1.18±1.29	1.19±1.41
9	1.28±1.09	1.29±1.30	1.30±1.44
10	1.39±1.09	1.40±1.31	1.41±1.46

4.2 Timeout

A timeout event occurs when a packet, or its acknowledgement, is delayed too long. Let RTT_k denote the roundtrip time experienced by packet k and its corresponding acknowledgement. The TCP algorithms estimates the mean and deviation of the roundtrip time. Let $\widehat{\text{RTT}}_k$ and $\hat{\sigma}_k$ denote the estimated roundtrip time and deviation, based on measurements up to RTT_k. TCP then computes the timeout value for the next packet as $\widehat{\text{RTT}}_k + 4\hat{\sigma}_k$, which means that the probability that packet k causes a timeout is given by

$$P_{\text{TO}} = P(\text{RTT}_k > \widehat{\text{RTT}}_{k-1} + 4\hat{\sigma}_{k-1})$$

We assume that the values RTT_k are identically and independently distributed according to the delay distribution given in Section 4.1. For simplicity, we also assume that the estimates $\widehat{\text{RTT}}_k$ and $\hat{\sigma}_k$ are perfect and equal to the true mean and standard deviation of RTT_k.

The value of P_{TO} will of course depend on the delay distribution. In TCP, timeout is the last resort recovery mechanism, and in order to get reasonable performance, a timeout must be a very rare event. Let us look at some examples:

- If RTT_k is uniformly distributed on an interval, then $P_{\text{TO}} = 0$.
- If RTT_k happen to be normally distributed, then timeouts will be rare: We get $P_{\text{TO}} \approx 1/(1.5 \cdot 10^4)$.
- For an arbitrary distribution with finite mean and variance, Chebyshev's inequality, $P(|X - \mu| \geq a\sigma) \leq 1/a^2$, yields the bound $P_{\text{TO}} \leq 1/16$.

We can expect the P_{TO} to lie somewhere between the extreme cases of a normally distributed delay, and the pessimistic value given by Chebyshev's inequality. When calculating P_{TO} from the delay distribution of Section 4.1, we get the probabilities in Figure 3.

We see that the probability for spurious timeout is significant for all packet sizes. The dependence on step size and packet size is complex, it seems to be peculiarities of the ARQ scheme, e.g. the "RLP NAK guard time", shining through.

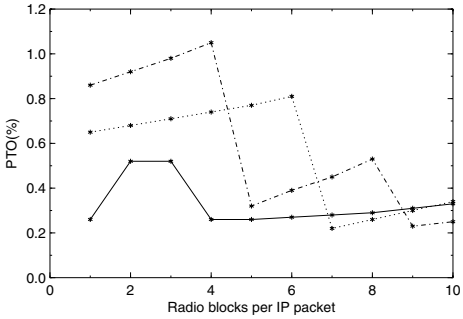

Fig. 3. Probability of spurious timeout. The solid line is for $\Delta = 0.2$, dotted for $\Delta = 0.06$ and dash-dotted for $\Delta = 0.02$.

5 Throughput Degradation

In this section, we derive an estimate for the performance degradation due to either of spurious fast retransmits and spurious timeouts. The expression for the relative degradation depends only on the probability of the event in question, the type of event, and the number of packets in the maximum congestion window. We illustrate the procedure by calculating the degradation for an example link.

When computing TCP throughput, there are two distinct cases: Depending on the bandwidth-delay product, throughput can be limited either by the bandwidth of the path across the network, or by the maximum TCP window size.

For a small bandwidth-delay product, a modest buffer before the bottleneck link (which we will assume is our radio link) will be enough to keep the radio link fully utilized. Timeouts and fast retransmit events, if they occur with a low frequency, will not affect throughput at all. This can be seen for example in the performance evaluation [16]: In the scenarios that have a large maximum window size compared to the bandwidth-delay product, we get a throughput that is the nominal radio link bandwidth times $1 - p$, and there is no significant difference between different link retransmission schemes. Only when bandwidth or delay is increased, or the maximum window size is decreased, do we see a drastic changes in throughput when the BLER or retransmission-scheme varies.

Therefore, we will concentrate on the case of a large bandwidth-delay product. For a concrete example, we will consider the following scenario: Radio link bandwidth 384 kbit/s, maximum TCP window size $w = 8192$ bytes, packet size $m = 1500$ bytes, and a constant roundtrip delay time, excluding the radio link itself, of 0.2 s. We will also consider a smaller packet size of $m = 960$ bytes, making each IP packet fit in a single radio block.

T will denote the mean end-to-end roundtrip time, 0.25 s for the larger packets and and 0.23 s for the smaller packets. This implies that we have the large bandwidth-delay product case, with an ideal throughput w/T of 32 Kbyte/s for the large packets and slightly larger for the smaller ones. Note that this is smaller than the available radio bandwidth of the link, $384000(1 - p)/8 \approx 42$ Kbyte/s

Table 2. Performance degradation.

Degradation	Packet size	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
due to P_{TO}	Small	5.6%	13.0%	16.5%
	Large	4.9%	6.4%	8.4%
due to P_{FR}	Small	2.5%	5.9%	7.3%
	Large	0.0%	0.1%	0.2%

Assume that a spurious timeout occurs independently with a small probability P_{TO} for each packet. Consider a typical cycle starting with a timeout event, followed by $1/P_{\mathrm{TO}}$ packets that are sent without timeouts or retransmissions. We compare the throughput during this cycle with the ideal throughput w/T. For simplicity, we use only congestion windows that are an integral number of packets, rounding down when needed, and we also assume that the cycle length $N = 1/P_{\mathrm{TO}}$ is an integer. The cycle can be divided into two phases: An initial recovery phase of r roundtrip times, where we send ℓ packets, followed by the second phase of $(N - \ell)m/w$ roundtrip times where we send at full speed, w/m packets each roundtrip time, $N - \ell$ packets in all.

The throughput during one cycle can be expressed as

$$\text{throughput} = \frac{N}{r + (N - \ell)m/w} \frac{m}{T} \tag{2}$$

Compared to the ideal throughput w/T, the relative degradation boils down to

$$\left(\frac{w}{T} - \text{throughput}\right) / \frac{w}{T} = \frac{d}{d + N} = \frac{1}{1 + 1/(dP_{\mathrm{TO}})} \tag{3}$$

where we have substituted $d = rw/m - \ell$, the number of additional packets we would have sent during the cycle if we had never decreased our congestion window.

5.1 Degradation due to Timeouts

Consider the effect of timeout. When recovering from timeout, the slowstart threshold is set to $w/2$. The congestion window is reset to one packet, and doubled each roundtrip time, until it reaches the slowstart threshold. Above the slow start threshold, the usual increment of the congestion window of one packet per roundtrip time is used until we get pack to the maximum value w.

For $m = 1500$, we take $w = 5m = 7500$ (so that $w/m = 5$ is an integer). The length of the recovery phase is 4 roundtrip times, during which we send 1, 2, 3 and 4 packets. We get $d = 10$, and from the P_{TO} calculated in section 4.2 we get a performance degradation of 4.9%, 6.4%, and 8.4% for our three step sizes.

For $m = 960$ we take $w = 8m = 7680$. The length of the recovery phase is 6 roundtrip times, during which we send 1, 2, 4, 5, 6, 7 packets. Thus, $d = 23$, and we get a performance degradation of 5.6%, 13.0% and 16.5% for the three step sizes.

5.2 Degradation due to Fast Retransmit

With large delay variations, packets can get so severely reordered that they trigger the TCP fast retransmit. The probability of spurious fast retransmit (P_{FR}) can be estimated from the loss/success-sequence probabilities. It turns out that unless the link is configured to do "in-order delivery", P_{FR} is significant for $n = 1$ ($0.25\% < P_{\mathrm{FR}} < 0.8\%$). P_{FR} decreases very rapidly with increasing n.

The TCP performance degradation for both spurious timeout and spurious fast retransmit is summarized in Table 2. We lose up to 7% due to fast retransmits, when using small packets, and up to 16% due to timeouts.

6 Conclusions

The properties of the lossy radio link considered in this paper reduces TCP throughput. By modeling the underlying processes controlling transmission power and retransmission scheduling on the link (Section 2 and 3), we can calculate the link properties, in particular the delay distribution, which are relevant for TCP (Section 4). This helps us getting a better understanding of the TCP behavior when communicating over modern wireless links.

In Section 5 we found that when using IP packets that fit in one or two radio blocks, as is common for high bandwidth wireless links, the link will generate spurious fast retransmits. Spurious timeouts will also occur, and this effect is significant for both large and small packets. The timeout value in TCP makes timeout a very rare event if the roundtrip time has a uniform or normal distribution, but that is not the case for the delay due to link layer retransmissions.

Some possible approaches to improved TCP performance over radio links are:

- Change the TCP algorithms to make them more robust to link irregularities.
- Engineer the link layer, to give it properties that plain TCP handles well.
- Aim for small delays, both over the wireless link and in the rest of the network, and for saturation of the wireless link.

Improvements of the TCP algorithms is an area of intense research. A drawback is that deployment of new algorithms affect all Internet end systems, which makes it a slow and costly process.

Tuning the link properties is more practical from a deployment point of view, at least if the tuning can be done before widespread adoption of a new link type. For example, spurious fast retransmits can be eliminated by having the wireless receiver buffer packets, making sure that packets are never forwarded out of order (this "in-order" option is already available on real systems [10]). To reduce the number of spurious timeouts, the link or either TCP end point could artificially increase the delay variation by adding an extra uniformly distributed delay to packets or acknowledgements. Also the link level retransmissions themselves can be seen as an example of changing the link properties to make the link more TCP friendly.

The point of reducing delay is that if the bandwidth-delay product is small enough that the radio link can be saturated by an ordinary TCP sender, then a

modest size buffer just before the radio link is enough to keep the link saturated, regardless of fluctuations in the TCP congestion window. It is hard to say if reducing bandwidth-delay product is a realistic objective, as it depends on the speed and delay of future link technologies. Extensions to TCP that increases the maximum window size would also help, for similar reasons [17].

References

1. Jacobson, V.: Congestion avoidance and control. In: Proc. of SIGCOMM. Volume 18.4. (1988) 314–329
2. Floyd, S.: TCP and explicit congestion notification. ACM Computer Communication Review **24** (1994) 10–23
3. DeSimone, A., Chuah, M., Yue, O.: Throughput performance of transport-layer protocols over wireless LANs. In: IEEE Globecom'93. (1993) 542–549
4. Xylomenos, G., Polyzos, G.C.: Internet protocol performance over networks with wireless links. IEEE Network **13** (1999) 55–63
5. Zorzi, R., Chockalingam, A., Rao, R.R.: Throughput analysis of TCP on channels with memory. IEEE J-SAC **18** (2000) 1289–1300
6. Mascolo, S., Casetti, C., Gerla, M., Sanadidi, M.Y., Wang, R.: TCP Westwood: bandwidth estimation for enhanced transport over wireless links. In: MobiCom, Rome, Italy (2001)
7. Ludwig, R., Katz, R.H.: The Eifel algorithm: Making TCP robust against spurious retransmissions. ACM Computer Communication Review **30** (2000)
8. Canton, A., Chahed, T.: End-to-end reliability in UMTS: TCP over ARQ. In: Globecom 2001. (2001)
9. Garrosa, P.M.: Interactions between TCP and channel type switching in WCDMA. Master's thesis, Chalmers University of Technology (2002)
10. Dahlén, A., Ernström, P.: TCP over UMTS. In: Radiovetenskap och Kommunikation 02. RVK (2002)
11. Gunnarsson, F., Gustafsson, F.: Power control in wireless communications networks — from a control theory perspective. Survey paper in IFAC World Congress, Barcelona (2002)
12. Sampath, A., Kumar, P.S., M.Holtzman, J.: On setting reverse link target SIR in a CDMA system. In: Proc. IEEE Vehicular technology conference. (1997)
13. Bai, Y., Rudrapatna, A., Ogielski, A.T.: Performance of TCP/IP over IS-2000 based CDMA radio links. In: Proc. of IEEE 52th VTC'2000-Fall, IEEE (2000)
14. Lundquist, H., Karlsson, G.: TCP with forward error correction. `http://www.it.kth.se/~hen/lundqvist_tcpfec.pdf` (2002)
15. Liu, B., Goeckel, D.L., Townsley, D.: TCP-cognizant adaptive forward error correction in wireless networks. `http://www-net.cs.umass.edu/~benyuan/pub/wirelessTCP.pdf` (2002)
16. Khan, F., Kumar, S., Medepalli, K., Nanda, S.: TCP performance over CDMA2000 RLP. In: Proc. IEEE 51st VTC'2000-Spring. (2000) 41–45
17. Jacobson, V., Braden, R., Borman, D.: TCP extensions for high performance. RFC 1323 (1992)