Analysis of the influence of power control and link level retransmissions on TCP performance over radio links

Niels Möller
Royal Institute of Technology
Email: niels@s3.kth.se

Karl Henrik Johansson
Royal Institute of Technology
Email: kallej@s3.kth.se

***Abstract*—**

I. INTRODUCTION

A fundamental assumption of the TCP protocol is that packet losses indicate congestion on the network [1], [2]. This is a problem when using TCP over wireless links, because a noisy radio transmission may erroneously indicate congestion and thereby reducing the TCP sending rate, e.g., [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, i.e. 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].

In general, it would be desirable for the end-to-end TCP to distinguish between packets that are delayed due to the retransmissions on the radio link and packets that are lost due to congestion on some wired link. Attempts to robustify TCP include the Eifel algorithm [7] and TCP Westwood [6]. The deteriorating effect on TCP of channel type switching for resource allocation in WCDMA was studied in [9]. Despite recent progress, the influence of the radio link properties on TCP is far from fully understood.

II. 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.

But there is no fix relationship between the SIR and the quality of the link. The quality measure we are interested in is the block error rate, BLER. To keep the BLER constant, an outer power control loop adjusts the target SIR of the inner loop, based on experienced block errors. The target BLER is

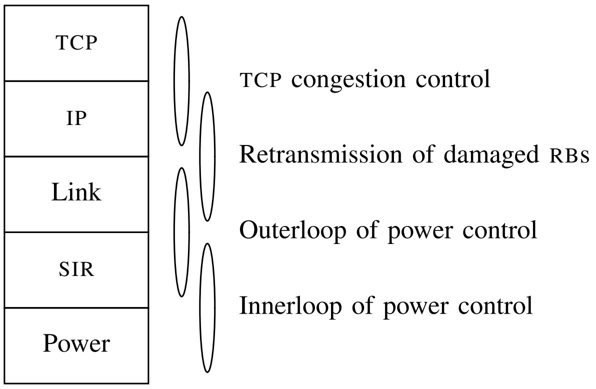

Fig. 1. System overview

deployment parameter, values between 0.1% and 10% have been used.

The simplest way to transmit IP packets over the wireless link is to package each IP packet into the appropriate number of radio blocks, and drop any IP packet where any of the corresponding radio blocks were damaged. This mode of operation may be appropriate for some applications, e.g. real time voice with error coding. 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 (that is needed for power control anyway), and it can use this information to request that damaged blocks be retransmitted over the link. This capability is an option in standard wireless network protocols, see [10] for an evaluation of these options in the IS-2000 and IS-707 RLP standards. Alternative approaches included 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, so that data from most lost packets can be recovered at the receiver. **FIXME: Reference?**

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

And finally, we have the end to end congestion control of TCP. The role of TCP is to adapt the sending rate to the available bandwidth. The input to the TCP algorithms is measured round trip time, and experienced loss events, both of which depends on the links traversed by the TCP packets. To understand and predict the behaviour of TCP, we must consider how it interacts with the control on the link, in particular the link level retransmissions, and the power control.

The control behaviour of TCP and of the power control outer loop are remarkably similar. This can be perhaps explained by the two facts that both operate close to a limit where the system performance degrades rapidly due to losses (for power control, it's losses due to noise and interference , for TCP, it's losses due to congestion), and both have very limited feedback, essentially a loss indication. The control, in both cases, consists of making small moves towards the limit as long as long as there is no indication of loss, and making a larger move in the other direction as soon loss is detected.

III. Radio model

In this section, we describe our model of the radio link, as it appears above the power control layer.

Data is transmitted over the radio link as a sequence of *radio blocks* (RB). One radio block corresponds to a time slot or *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 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 transmitters are controlled, so that the loss probability stays fairly constant. The reference 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 [11]. In the following, we thus assume $p = 0.1$.

A. Power control

The typical power control consists of an inner loop that tries to keep the signal to interference ratio (SIR) close to a reference value. This loop typically have 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 [12], 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 the reference SIR. This loop uses feedback from the decoding process, usually the bit error rate (BER) or block error rate (BLER). 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 rates are small, one approach is to increase the reference SIR significantly when an error is detected, and decrease the reference SIR sligtly for each block that is received successfully. It is interesting to see how this strategy resembles the TCP "additive increase, multiplicative decrease" flow control strategy.

For a survey of modern power control techniques for systems such as WCDMA, see [12].

B. Markovmodel

The outer loop of the power control sets the reference value for the SIR. Given a particular reference value r, the received 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, we the probability of block errors is a function $f(r)$ of the reference SIR.

One can compute $f(r)$ from models for the channel and coding. The BPSK example given in [13]

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

where α is the coding gain, $\beta = \ln 10/10$, σ is the standard deviation of the received SIR, and N is the number of bits in a radio block. We take $alpha = 4$, from [13], $sigma = 1dB$, corresponding to a small SIR oscillation as in [12], and $N = 2560$, corresponding to a modest link capacity of 128 kbit/s.

However, for control purposes, even a crude approximation of the true $f(r)$. 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 interesting area is the transision in between, and the power control will operate close to the point $f(r) = p = 0.1$.

The outer loop of the power control uses discrete values for the reference SIR. One mechanism to keep the block error probablity close to desired probability p, uses a fixed step size Δ as follows. Whenever a radio block is received successfully, the reference SIR for the next block is decreased by Δ. And whenever a radio block is damaged, the reference SIR for the next block is increased by $K\Delta$, where $1/(1 + K) = p$. The value of Δ is an important control parameter, that determines the performance of the power control. However, we will see that the stationary behaviour of the loss process is mostly independent of Δ.

For an integer K, in our case $p = 0.1$ implies $K = 9$, the varying reference SIR can be viewed as a discrete Markov chain [13]. To do this, we have to make the assumption that block errors depend only on the reference SIR when the block is transmitted, there's no relevant history except the SIR.

For a given function $f(r)$ and a step size Δ, we 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 probabilities.

Figure 2 shows the threshold function $f(r)$, together with the stationary distribution for three different stepsizes. Note that 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 reaction times. **FIXME: What's the proper terminology?**

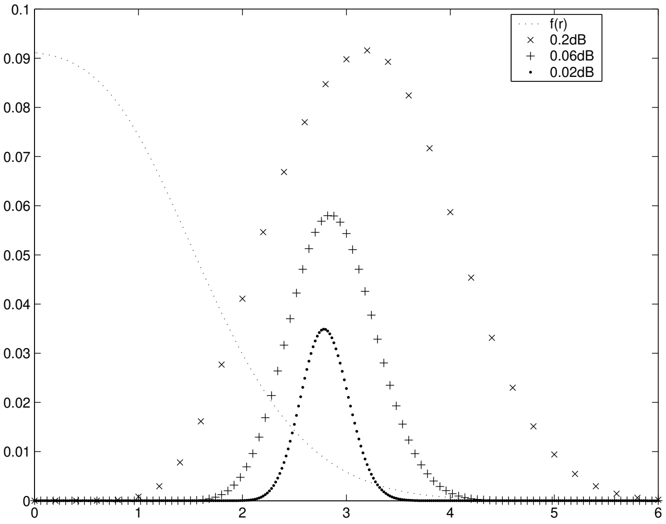

Fig. 2. Stationary distribution for the power control. The dotted curve is the threshold-shaped function $f(r)$, scaled to fit in the figure. The dots represent the stationary distribution for three different stepsizes.

IV. Link-layer retransmissions

There are several schemes for link level retransmission. We will consider one of the simpler, the (1,1,1,1,1)-Negative Acknowledgement scheme [14], 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. Number the time slots after this time starting from one. The probability of any particular finite loss/success sequence (for example, the first block (k = 1) was damaged, the next six were received successfully, the eigth damaged) can be calculated by conditioning on the start state. We have done an exhaustive calculation of these probabilities for all loss/success sequences of length up to 20. From these numbers, one can investigate the experience of an IP packet, or sequence of IP packets, that traverses the link.

Before going into details, let us first consider the influence of the step size Δ. One way to measure if Δ is "small" or "large" is to count the number of states with a loss probability $0.05 < f(r) < 0.5$, i.e. within a factor of two from our target BLER. It turns out that if this number, call it M, is at least a few times larger than 10 (remember that the upward jumps in reference SIR are 9 states), say at $M > 70$, corresponding to a "small" Δ, then the retransmission behaviour is mostly independent of the step size. This can be understood as a choice of parameters where a linearization around the operating point

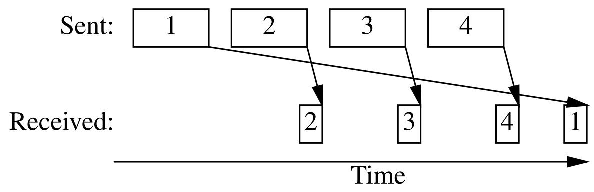

Fig. 3. Reorder triggers fast retransmit

$f(r) = p = 0.1$ is a good approximation.

V. TCP/IP properties

A. IP *packet delay*

One important feature, in particular for TCP, is the delay distribution of packets traversing the link. The full distribution can be computed explicitly. The expected value and standard deviation for three values of the step size is shown in table I.

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

TABLE I

Mean and standard deviation of the delay distribution

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 nTTI). Comparing the delay distribution for small and large stepsizes, there are significant differences in the individual probabilities, for example the probability that a tree-block packet experiences a one TTI delay due to retransmissions, however this mostly affects the variance, not the mean.

The clearest influence of the step size can be seen when considering the probability that a large IP packet, $n = 10$, does not experience any retransmissions. If such a transmission is successful, the SIR is decreased by Δ for each of the ten blocks sent, which makes success highly unlikely if Δ is large. But as $\Delta \to 0$, the probability will approach $(1-p)^10 \approx 35\%$. For our three example step sizes, that probability is 16%, 27%, and 32%, where the largest value (corresponding to $\Delta = 0.02$) is fairly close to the limit.

B. TCP *fast retransmission*

If a packet in a TCP stream is delayed so much that it lags behind three other packets (see Figure 3), the acknowledgements for the next three packets are "duplicated", and the sender will go into fast retransmit/fast recovery mode. That means that from the point of view of TCP and its congestion control, the packet is lost.

Computing the precise number of fast retransmit events is not trivial. However, we can estimate the probability that fast retransmits occur as follows. Consider a randomly chosen time, where the power control state is governed by the stationary distribution. Assume that four IP packets, each of size n radio blocks, arrives to the radio link back-to-back. From our earlier probability computations, we can derive the probability that the first packet is delayed long enough to trigger a fast retransmit at the receiving TCP, shown in table II. Here, we see significant differences between small and large step sizes.

n	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
1	0.26%	0.63%	0.79%
2	0.00%	0.02%	0.04%
3	0.00%	0.00%	0.00%

TABLE II

PROBABILITY OF SPURIOUS FAST RETRANSMIT

The probability decreses rapidly as the packet size increases. Therefore, spurious fast retransmits will not be a problem for larger packet sizes.

C. Timeout

A timeout event occurs when a packet, or its acknowledgement is delayed too long. Let RTT_k denote the roundtrip time (i.e. the delay experienced by a data packet and its corresponding acknowledgement) k. The TCP algorithms estimates the roundtrip time and the variation thereof, let $\widehat{\text{RTT}}_k$ and $\hat{\sigma}_k$ denote the estimated roundtrip time and variation, 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 the values RTT_k are identically and independently distributed according to the delay distribution given in section V-A. For simplicity, we will 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's look at some examples. If RTT is uniformely 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 we can apply Chebysev's inequality of probability, $P(|X - \mu| \geq a\sigma) \leq 1/a^2$, to get the bound $P_{\text{TO}} \leq 1/16$.

We can expect the P_{TO} to lie somewhere between those two extreme cases. When calculating P_{TO} from the delay distribution calculated in section V-A, we get the numbers in table III.

We see that unlike spurious fast retransmits, the probability for spurious timeouts is significant for all packet sizes.

n	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
1	0.26%	0.65%	0.86%
2	0.52%	0.68%	0.92%
3	0.52%	0.71%	0.98%
4	0.26%	0.74%	1.05%
5	0.26%	0.77%	0.32%
6	0.27%	0.81%	0.39%
7	0.28%	0.22%	0.45%
8	0.29%	0.26%	0.53%
9	0.31%	0.30%	0.23%
10	0.33%	0.34%	0.25%

TABLE III

PROBABILITY OF SPURIOUS TIMEOUT

When the TCP sender detects a timeout, it reacts by setting the slowstart threshold to half the current window size, and resetting the window size to one packet. Slow start, increasing the congestion window by one packet per acknowledgement is used until the congestion window reaches the slow start threshold, after which it enters ordinary congestion avoidance mode, increasing the congestion window by one packet per RTT.

It may be possible for the link or the receiver to reduce the timeout probability be adding an artificial delay to packets or acknowledgements. If a normal distributed delay with variance of a few TTIs is added, that will inflate the sender's estimate of the RTT variance, as well as the timeout value.

D. Throughput degradation

In this section, we derive an estimate for the performance degradation due to either spurious fast retransmits and spurious timeouts. The expression for the relative degradation is simple, it depends only on the probability of the event in question, the type of event, and the number of packets in a the maximum congestion window. We also illustrate the procedure by calculating the degradation for an example link. We will see that we lose up to 7% due to fast retransmits, when using small packets, and up to 16% due to timeouts. See table IV for all the numbers.

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

TABLE IV

PERFORMANCE DEGRADATION

When computing TCP throughput, there are two distinct cases: Throughput can be limited either by the bandwidth of the path across the network, or by the maximum TCP window size. Which case depends on the bandwidth-delay product of the network, and on the sender's and receiver's window limits.

For a small bandwidth-delay product, a modest buffer before the bottle-neck link (which we'll usually assume is our radio link) will be enough to keep the radio link fully utilized, and the throughput will be the nominal radio link bandwidth minus protocol overhead and retransission overhead (where the latter corresponds to the factor $1-p$). 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 Khan, Kumar et al, 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's no significant difference betwen 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 look at the following scenario: Radio link bandwidth 384 kbit/s, maximum TCP window size $w = 8192$ bytes, packet size $m = 1500$ bytes (typical for TCP bulk transfers), and a roundtrip delay time, excluding the radio link itself, of constant 0.2 s. We will also consider a smaller packet size of $m = 960$ bytes, making each IPpacket 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

Fast retransmit followed by fast recovery and timeout followed by slow start can be treated in a uniform way. Assume that fast retransmit (or timeout) occurs independently with a small probability p. Consider a typical cycle starting with the event that causes the performance degradation, followed by $1/p$ 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 congenstion windows that are an integral number of packets, rounding down when needed, and we also assume that the cycle length $N = 1/p$ is an integer. The cycle can be divided into two phases: An initial recovery phase of r roundtrip times, when we send ℓ packets, followed by the second phase of $(N-\ell)m/w$ roundtrip times when we send at full speed, w/m packets each roundtrip time.

The throughput during one cycle can be expressed as

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

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

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

where we have substituted $d = rw/m - \ell$, the difference between the number of packets we could have sent in the recovery phase using the maximum window size, and the number of packets we actually sent in that phase. Or equivalently, the number of additional packets we would have sent during the cycle if we had never decreased our congestion window.

E. Degradation due to fast retransmit

We start with the fast recovery. Here, the sending TCP will resend the delayed packet (thinking that it is lost), and halve its congestion window. The window size is increased by m each RTT, until it reaches the maximum value, w. For simplicity, we use only congenstion windows that are an integral number of packets, rounding down when needed. For $m = 1500$, we take $w = 5m = 7500$ (so that $w/m = 5$ is an integer).

When performing fast recovery, the congestion window is halved to $2m \approx w/2$, and during the next three roundtrip times we send 2, 3, 4 packets, or 9 packets total, after which we're up to full speed again, sending 5 packets each roundtrip time. This means that $x = 9$, $r = 3$ and $d = 6$: Due to the spurious fast retransmit event, we send 6 packets less during the cycle. So we get a performance degradation of $1/(1 + 1/(6P_{\text{FR}}))$. As each packet corresponds to two radio blocks, $n = 2$ in the table in section V-B, and we get a significant P_{FR} only for the smaller step sizes. $P_{\text{FR}} \approx 0.02\%$ gives $N = 5000$ and a performance degradation of $6/5006 \approx 0.1\%$. $P_{\text{FR}} \approx 0.04\%$ gives a oerformance degradation of $6/2506 \approx 0.2\%$.

For the packet sizes $m = 960$, the situation is different. We take $w = 8m = 7680$. During the 4 roundtrip times of the recovery phase, we send 4, 5, 6 and 7 packets, giving $d = 10$. For the large step size, we have $P_{\text{FR}} \approx 0.26\%$, which gives $N = 385$ and a performance degradation of $10/395 \approx 2.5\%$. Similarly, for the smaller step sizes, $P_{\text{FR}} \approx 0.63\%$ implies a performance degradation of about 5.9% and $P_{\text{FR}} \approx 0.79\%$ implies a performance degradation of about 7.3%.

F. Degradation due to timeouts

Let us next 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 $m = 1500, w = 5m$ the length of the recovery phase is 4 roundtrip times, during which we send 1, 2, 3 and 4 packets (we don't see any increments larger than one packet due to the small slow start threshold). We get $d = 10$, and from the P_{TO} table in section V-C we get a performance degradation of 4.9%, 6.4%, and 8.4% for our three stepsizes.

For $m = 960, w = 8m$ 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 stepsizes.

The numbers are summarized in table IV. We see that both spurious fast retransmits and spurious timeouts cause a significant decrease in throughput.

VI. Conclusions

The properties of the lossy radio link considered in this paper reduces TCP throughput. By modelling the underlying processes controlling transmission power and retransmission scheduling on the link, we can calculate the link properties, in particular the delay distribution, which are relevant for TCP.

We find that as soon as the step size used in the power control is reasonably large, the TCP performance properties (in a stationary state of the power control) are mostly decoupled from the performance properties of the power control, such as convergence speed and energy efficience.

We find that when using IP packets that are small compared to the radio block size, as is common for high bandwidth wireless links, the link will generate spurious fast retransmits at the TCP sender.

Spurious timeouts will also occur, and this effect is significant for both large and small packets. The timeout value in TCP makes timeouts 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.

To improve performance of TCP over radio links, there are several possible approaches.

- Change the TCP algorithms to make them more robust to link irregularities.
- Tune the parameters of the link, 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. **FIXME: Need reference**

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 option is already available on real systems **FIXME: Need reference**. To reduce the number of spurious timeouts, the wireless receiver or the TCP receiver could add an extra random delay to either packets or acknowledgements. Also the link level retransmissions themselves can be seen as an exaxmple 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 but 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's hard to say if reducing bandwidth-delay product is realistic goal, as it depends on the speed and delay of future link technologies.

For the first two approaches, it is crucial to model and understand the properties of the underlying link.

REFERENCES

[1] V. Jacobson, "Congestion avoidance and control," in *Proc. of SIGCOMM*, 1988, vol. 18.4, pp. 314–329.

[2] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, no. 5, pp. 10–23, 1994.

[3] A. DeSimone, M. Chuah, and O. Yue, "Throughput performance of transport-layer protocols over wireless lans," in *IEEE Globecom'93*, 1993, pp. 542–549.

[4] G. Xylomenos and G. C. Polyzos, "Internet protocol performance over networks with wireless links," *IEEE Network*, vol. 13, no. 4, pp. 55–63, 1999.

[5] R. Zorzi, A. Chockalingam, and R. R. Rao, "Throughput analysis of tcp on channels with memory," *IEEE J-SAC*, vol. 18, no. 7, pp. 1289–1300, 2000.

[6] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: bandwidth estimation for enhanced transport over wireless links," in *MobiCom*, Rome, Italy, 2001.

[7] R. Ludwig and R. H. Katz, "The eifel algorithm: Making tcp robust against spurious retransmissions," *ACM Computer Communication Review*, vol. 30, no. 1, January 2000.

[8] A. Canton and T. Chahed, "End-to-end reliability in umts: Tcp over arq," in *Globecom 2001*, 2001.

[9] P. M. Garrosa, "Interactions between tcp and channel type switching in wcdma," M.S. thesis, Chalmers University of Technology, 2002.

[10] Y. Bai, A. Rudrapatna, and A. T. Ogielski, "Performance of tcp/ip over is-2000 based cdma radio links," in *Proc. of IEE 52th VTC'2000-Fall*. IEEE, 2000.

[11] A. Dahlén and P. Ernström, "Tcp over umts," in *Radiovetenskap och Kommunikation 02*, 2002, RVK.

[12] Fredrik Gunnarsson and Fredrik Gustafsson, "Power control in wireless communications networks — from a control theory perspective," .

[13] Ashwin Sampath, P. Sarath Kumar, and Jack M.Holtzman, "On setting reverse link target sir in a cdma system," in *Proc. IEEE Vehicular technology conference*, May 1997.

[14] F. Khan, S. Kumar, K. Medepalli, and S. Nanda, "Tcp performance over cdma2000 rlp," in *Proc. IEEE 51st VTC'2000-Spring*, 2000, pp. 41–45.