43rd IEEE Conference on Decision and Control
December 14-17, 2004
Atlantis, Paradise Island, Bahamas

Making retransmission delays in wireless links friendlier to TCP

Niels Möller, Karl Henrik Johansson and Håkan Hjalmarsson
{niels|kallej|hjalmars}@s3.kth.se

Abstract— **Heterogeneous communication networks with their variety of application demands, uncertain time-varying traffic load, and mixture of wired and wireless links pose several challenging problem in modeling and control. In this paper we focus on the packet delay, which is an important variable for efficient end-to-end congestion control. In particular, we study the delay effects of radio links which use power control and link-layer retransmissions.**

Link-layer retransmissions induce delays which do not conform to the assumptions on which the transport protocol is based. This causes undesired TCP control actions which reduce throughput. A link-layer solution based on adding carefully selected delays to certain packets is proposed to counteract this problem. All information needed for this is available locally at the link.

I. Introduction

Congestion control is one of the key components that has enabled the dramatic growth of the Internet. The original idea [1] was to adjust the transmission rate based on the loss probability. The first implementation of this mechanism, denoted TCP Tahoe, was later refined into TCP Reno. This algorithm (together with some of its siblings) is now the dominating transport protocol on the Internet. The throughput and delay experienced by individual users depend on several factors, including the TCP protocol, link capacity, and competition from other users. As illustrated in Figure 1, there are also lower layers that may affect the achieved delay and bandwidth, if part of the end-to-end connection is a wireless link.

Poor TCP performance over wireless links is a well-known problem. The traditional explanation for poor TCP performance is that the wireless link drops packets due to noise on the radio channel, and that TCP interprets all packet losses as indications of network congestion. This explanation is a little too simplistic when considering wireless links that employ link-layer retransmissions, which is common in cellular systems. The link-layer retransmission scheme transforms a lossy link, with fairly constant delay, into a link with few losses but random delays. The resulting delays, it turns out, are also problematic for TCP [10].

Several approaches to improve wireless TCP behavior have been suggested in the literature. Modifications of TCP have been proposed [2], [3]. Other methods try to more directly differentiate loss as being either due to congestion or due to lossy wireless transmissions [4], [5], [6]. Performance-enhancing proxies is an alternative in which either split connection schemes or interception schemes are used. The first approach introduces a virtual user at the link which acts as receiver to the source and source to the receiver. In the latter approach, acknowledgments are monitored and dropped if they indicate packet loss due to link-layer retransmissions. Finally, it is possible to counteract the influence from the wireless link by letting the receiver control the transmission via its advertised window. See [7], [8] for further details on these schemes.

This work was supported by European Commission through the project EURONGI and by Swedish Research Council.

We believe that, as far as possible, the link-layer should be engineered to be TCP-friendly, reducing the differences between wired and wireless links. The main contribution of this paper is an optimization procedure that improves TCP performance by adding carefully selected delays to certain packets. There will naturally be some residual idiosyncrasies of wireless channels that cannot be dealt with in the link-layer; our approach should be viewed as complementing both developments to make TCP more robust to "strange" links, and cross-layer developments that let the link and the end-nodes exchange information about link and flow properties.

This article is organized as follows. Section II describes our models for the lower layers of the system. In Section III we use these models to derive IP-level properties, in particular the IP-packet delay distribution, and implications for TCP performance. In Section IV we argue that we should use the engineering freedom we have in the link-layer to make the radio link more friendly to TCP, and we describe our delay optimization procedure and its impact on TCP.

II. System Model

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 layer, 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 varying radio conditions. Next, we have an outer power control loop that tries to keep the block error rate (BLER) constant, by adjusting the target SIR (measured in dB) of the inner loop.

On top of power control, we have local, link-layer, retransmissions of damaged blocks. Finally, we have the end-to-end congestion control of TCP.

By modeling the lower layers, we can investigate the effects the link layer control have on TCP performance. We

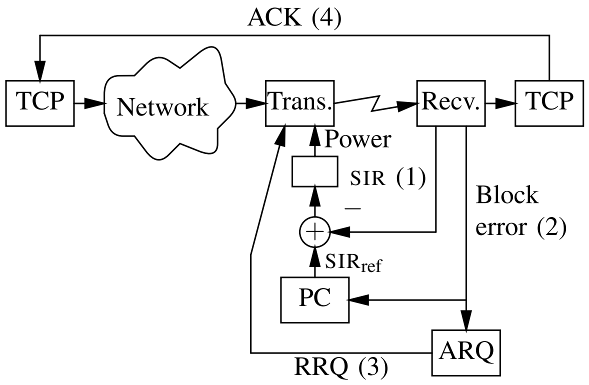

Fig. 1. End-to-end congestion control is affected by the delay and the bandwidth of the wired part of the network, but also by feedback mechanisms in lower layers of the wireless links.

refer to our previous paper [10] for further details on the radio model.

A. Power control Markov chain

The objective of the power control is to keep the block error rate constant. The target BLER 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 [9], which is what we will use. As there is no simple and universal relationship between the SIR and the block error rate, the outer power control loop uses feedback from the decoding process to adjust SIR_{ref}. The outer loop uses a fixed step size Δ. It decreases SIR_{ref} by Δ for each successfully received block, and increases SIR_{ref} by 9Δ each time a block is damaged.

This process can be modeled as a discrete Markov chain, where state k corresponds to $\text{SIR}_{\text{ref}} = k\Delta$. Assuming that the inner loop power control manages to keep the actual SIR close to SIR_{ref}, and using an appropriate channel model, we get a threshold shaped function $f(r)$ which gives the probability of block damage for any $\text{SIR}_{\text{ref}} = r$. There are two transitions from state k of the Markov chain: To state $k + 9$, with probability $f(k\Delta)$, and to state $k - 1$ with probability $1 - f(k\Delta)$. The operating point of the outer loop power control is close to the point where $f(r) = 10\%$, i.e., the desired block error rate.

From $f(r)$ and Δ, it is straightforward to compute the stationary distribution of the Markov chain. Figure 2 shows the stationary distribution for a BPSK channel (see [10] for the parameters) for three different values for Δ.

B. Link-layer retransmission

Since a packet loss probability on the order of 10% would be detrimental to TCP performance, the link detects block damage (this is the same feedback signal that is used for the outer loop power control), and damaged blocks are scheduled for retransmission. We will consider one simple retransmission scheme, the (1,1,1,1,1)-Negative Acknowledgment scheme [11], 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 is 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.

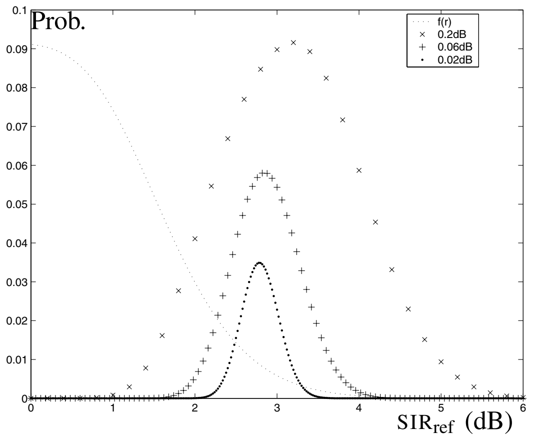

Fig. 2. Stationary distribution for the power control. Each mark represents one state of the power control, the corresponding value of SIR_{ref}, and its stationary probability. The dotted curve is the threshold-shaped function $f(r)$, scaled to fit in the figure, which represents the block error probability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

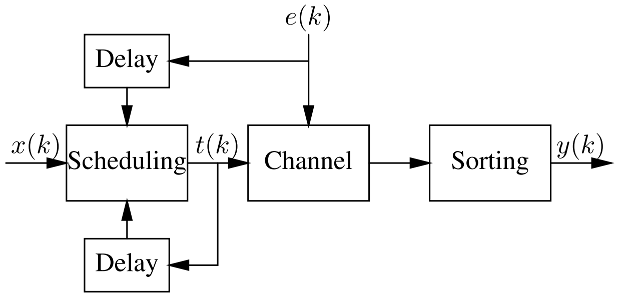

Fig. 3. Retransmission model

C. Retransmission as feedback

To be able to analyze the impact of the scheduling mechanism on link properties such as the delay distribution, it is of interest to model the retransmission scheme. Feedback is an intrinsic property of the retransmission mechanism. Below we propose a model for the relationship between the input blocks, the block error process, and the in-order output blocks, where this feedback is explicitly shown. We believe that this model will be useful for further studies of retransmission scheduling.

Let k denote time in units of the transmission time interval (TTI), and consider the following input and output signals, also shown in Figure 3.

$$x(k) = \text{\# of input blocks up to time } k$$
$$e(k) = \text{\# of errors up to time } k$$
$$y(k) = \text{\# of in-order output blocks up to time } k$$

These are accumulated rate functions, also used in network calculus, and increasing. Let

$$t(k) = \text{Index of block transmitted at time } k$$

which is *not* an increasing function. Consider a simple one-parameter family of retransmission schemes, where each damaged block is retransmitted g slots later, and there is no limit on the number of times a block may be resent. The parameter g corresponds to the RLP NAK guard time.

To describe the process mathematically, we start with the queue at the input to the scheduler. Let $s(k)$ be the number of time slots up to time k that are not used for retransmissions, and let $f(k)$ be the number of blocks that have been transmitted (but not necessarily received successfully) up to time k. Then

$$s(k) = k - e(k-g)$$
$$f(k) = \min_{\ell \le k} (x(\ell) + s(k) - s(\ell))$$

where the minimum in the latter equation is obtained when ℓ is the start of the current busy period.

The scheduling can be described as

$$t(k) = \begin{cases} f(k) & \text{if } e(k-g) = e(k-g-1) \\ t(k-g) & \text{if } e(k-g) > e(k-g-1) \end{cases} \quad (1)$$

The first case corresponds to an original transmission, and the second case to a retransmission. Finally, $y(k)$ is defined by $y(k) = n$ if all blocks up to n have been received properly at time k, but block $n+1$ has not. In symbols,

$$y(k) = \max\{n : \forall \ell \le n, \exists j \le k, t(j) = \ell, e(j) = e(j-1)\}$$

So where is the feedback? It is included explicitly, in Equation 1. This model, together with a model for the stochastic process $e(k)$, lets us optimize the parametrized retransmission scheme. The delay at time k is defined by

$$d(k) = \min\{\tau \ge 0 : y(k+\tau) \ge x(k)\}$$

If x and e are stationary processes, with average rates that sum to less than one, then also $d(k)$ is a stationary process, and its properties can, at least in principle, be calculated from x, e, and the retransmission model. If $Q(d)$ is a quality measure that depends on the properties of d, we can formulate the optimization problem $g^* = \arg\max_g Q(d)$ which gives the optimal value for the retransmission delay.

Intuitively, we expect that g^* will depend on the autocorrelation of e; it seems reasonable to use a retransmission delay such that the correlation, between loss of the original transmission and loss of the retransmission, is small.

This one parameter retransmission model is quite limited. Other schemes can be modeled analogously, as long the relation between e and s is simple, and the scheme does not need an additional queue for retransmitted packets. The challenge is to find a powerful but simple parameterization of an interesting class of schemes.

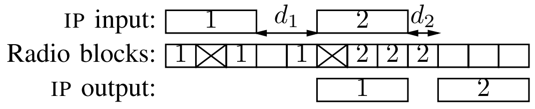

Fig. 4. IP packets divided into radio blocks

III. TCP/IP Implications

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 second and the sixth block damaged, the rest received successfully), we can calculate the probability by conditioning on the initial power control state and following the corresponding transitions of the Markov chain. We can then use these probabilities to investigate the experience of IP packets traversing the link.

A. IP *packet delay*

As a link employing link-layer retransmission yields a very small packet loss probability, the most important characteristic of the link is the packet delay distribution. If the distribution is sufficiently "friendly" to TCP, then the layering of the system works nicely, which means that upper layers like TCP need not be aware of any particular properties of individual links in the network.

We first compute the packet delay distribution explicitly from the models described above. Later, we will also assume that the calculated delay probabilities apply independently to all packets, which should be fairly close to reality as long as the power control is working.

When transmitting variable size IP packets over the link, each packet is first divided into fixed size radio blocks, see Figure 4. Let n denote the number of radio blocks needed for the packet size of interest. Typically, $1 \le n \le 10$.

The delay experienced by an IP packet depends on which, if any, of the corresponding radio blocks are damaged, and on the number and scheduling of the block retransmissions. When all the blocks are finally received correctly, the IP packet can be reassembled and passed on.

Delay (ms)	0	40	60	100	120	160	180
Probability (%)	80.6	8.8	9.3	0.6	0.6	0.03	0.03

TABLE I

From the probabilities for all possible success/loss sequences at the radio block level, we can extract explicit probabilities for the possible IP packet delays. The resulting delay distribution for our example channel (see Figure 5), with a power control step size $\Delta = 0.06$ dB and $n = 2$ is shown in Table I. This table includes only the delays due to radio block retransmissions, there is also a fixed delay of 40 ms for the transmission of two radio blocks.

B. TCP *performance degradation*

In observations and performance evaluations of TCP over wireless links [11], the properties of a wireless link can shine through to the TCP layer in three different ways:

- *Genuine packet loss:* With bad enough radio conditions, packet drops are inevitable. We will not consider such genuine packet loss here, as we assume that the radio channel is good enough that the power control and link-layer transmissions can get packets through.
- *Packet reorder:* For a link with highly variable delay, packets can get reordered. Severe reordering can trigger a spurious TCP fast retransmit.
- *Spurious timeout:* A packet that is not lost, only severely delayed, can trigger a spurious TCP timeout.

If the product of the end-to-end roundtrip delay and the available bandwidth is small, compared to the maximum TCP window size, spurious timeouts and spurious fast retransmit need not lead to any performance degradation. A modest buffer before the radio link will be enough to keep the link busy even when the sender temporarily decreases its sending rate. On the other hand, if the bandwidth delay product is larger than the maximum TCP window size, throughput is decreased. The difference between these two cases can be seen for example in the performance evaluation [11]: 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 $1 - p$ (where p is the average block loss probability), 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 drastic changes in throughput when the BLER or retransmission-scheme varies.

We therefore concentrate on the case of a large bandwidth–delay product. Then both spurious fast retransmit and spurious timeout leads to a degradation of TCP performance, and we will consider them in turn.

C. *Spurious fast retransmit*

Spurious fast retransmit has been observed as an important factor in poor TCP performance over wireless links. The probability of spurious fast retransmit can be estimated from the loss/success-sequence probabilities. It turns out that unless the link is configured to do "in-order delivery", the probability is significant for $n = 1$ (0.25%–0.8%). It decreases rapidly with increasing n.

Fortunately, this problem is easy to solve: Let the radio link receiver sort packets so that they are always passed on in order. Typically, this is an option in the configuration of radio link equipment, and it should be enabled on links where TCP performance is important. (One may want to use separate channels for TCP packets and real time streams, and enable in-order delivery only for the TCP-packets.)

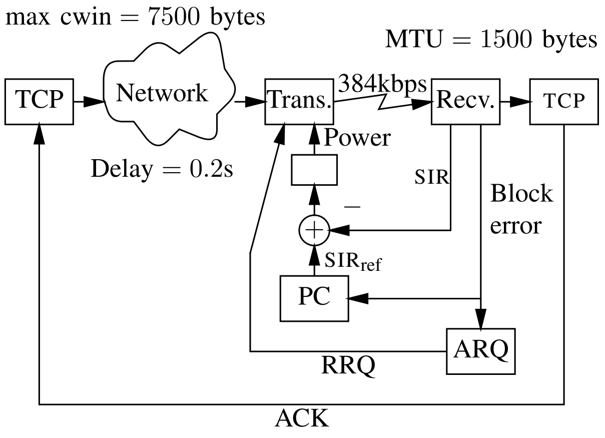

Fig. 5. Numerical example

D. *Spurious timeout*

A TCP timeout event occurs when a packet, or its acknowledgment, is delayed too long. Let RTT_k denote the round-trip time experienced by packet k and its corresponding acknowledgment. The TCP algorithm estimates the mean and standard deviation of the round-trip time. Let $\widehat{\text{RTT}}_k$ and $\hat{\sigma}_k$ denote these estimates, based on measurements up to RTT_k. TCP then computes the retransmission timeout value for the next packet as $\text{RTO} = \widehat{\text{RTT}}_k + 4\hat{\sigma}_k$, and the probability that packet k causes a spurious timeout is given by

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

An idealized model of TCP is to assume that the estimation is perfect, and that the timeout value is set to $\mu(\text{RTT}) + 4\sigma(\text{RTT})$, where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation. Then the spurious timeout probability is given by

$$\mathrm{P}_{\text{TO}}(\text{RTT}) = \mathrm{P}(\text{RTT} > \mu(\text{RTT}) + 4\sigma(\text{RTT})) \tag{2}$$

Note that P_{TO} is invariant under addition of constant delays.

From the delay distribution of Table I, we get $\text{RTO} \approx$ 103 ms and the probability that the delay is larger is $\mathrm{P}_{\text{TO}} \approx$ 0.68%. When varying the parameters n and Δ, we typically get a probability of spurious timeout on the order of 0.5%–1% [10]. This probability might seem small, but as shown next, it can have significant performance implications.

E. *Performance implications*

As explained above, we concentrate on the case of a large bandwidth–delay product. We will also assume that in-order delivery is enabled, so there is no spurious fast retransmit, only spurious timeout. For a concrete example, consider the scenario in Figure 5: radio link bandwidth 384 kbit/s, packet size $m = 1500$ bytes, maximum TCP window size $w = 7500$ bytes (i.e., five packets), and a constant round-trip delay time, excluding the radio link itself, of 0.2 s.

The available radio bandwidth (excluding the 10% lost radio blocks) is 42.2 Kbyte/s. Due to the limited window size, TCP cannot utilize the link fully. The ideal TCP throughput is one maximum size window per RTT. For the untweaked link, the mean total roundtrip time $\mu(\text{RTT})$ is $200 + 40 + 10.6 = 250.6$ ms, implying an ideal throughput of 29.2 Kbyte/s.

For each spurious timeout, the sending TCP enters slow start. The window size is reset to 1 packet, and the slowstart threshold is set to 2 packets. For the next four round-trip times, we will send 1, 2, 3, and 4 packets, i.e., 10 packets less than if we had kept sending a maximum window of 5 packets every RTT. This leads to

$$\text{Throughput} = \frac{w}{\mu(\text{RTT})(1 + 10\text{P}_{\text{TO}}(\text{RTT}))}$$

(a more general formula is derived in [10]). Hence, over the example radio link, we get a throughput of 27.4 Kbyte/s.

IV. Improving the Link-Layer

It is not trivial to define precisely what properties a link should have in order to be friendly to TCP. It seems clear that for example links with normal or uniformly distributed and independent delays are friendly enough. We define a measure of TCP-unfriendlyness by applying Equation 2 to an arbitrary stochastic variable X, representing the independent and identically distributed packet delays. For convenience, we also define $\text{RTO}(X)$ as the corresponding timeout value.

$$\begin{aligned}\text{RTO}(X) &= \mu(X) + 4\sigma(X)\\ \text{P}_{\text{TO}}(X) &= \text{P}(X > \text{RTO}(X))\end{aligned}$$

For some distributions of interest, we find

$$\begin{aligned}
&X \text{ uniform} &&\Longrightarrow \text{P}_{\text{TO}}(X) = 0\\
&X \text{ normal} &&\Longrightarrow \text{P}_{\text{TO}}(X) \approx 6.3 \cdot 10^{-4}\\
&X \text{ wireless} &&\Longrightarrow \text{P}_{\text{TO}}(X) \sim 100 \cdot 10^{-4}\\
&X \text{ arbitrary} &&\Longrightarrow \text{P}_{\text{TO}}(X) \leq 625 \cdot 10^{-4}
\end{aligned}$$

The last value is derived from Chebyshev's inequality, assuming only that X has finite first and second moments. We see that the first two distributions, which we know are friendly to TCP, yield a P_{TO} at least two orders of magnitude below the worst case given by Chebyshev. The wireless delay yields a significantly higher P_{TO}, although still with some margin to the worst case.

The motivation for this measure is the calculation of the timeout value in TCP. Timeout is intended to be the last resort recovery mechanism, and for TCP to work properly, spurious timeout must be a rare event. We therefore define a *TCP-friendly link* to mean a link with no loss or reorder, and with a delay distribution that yields a small P_{TO}.

If we want to improve the system, where should we put the effort? The power control design has many constraints of its own, relating to radio efficiency and cost of deployment. It seems difficult to design and motivate changes to the power control for improving the delay distribution properties. Improvements to the TCP algorithm in the end-nodes are important, but also difficult both for technical and practical reasons, such as limited information about what goes on in the link (note that the link and the TCP implementations are not only in separate layers, they are also *geographically* separate), and the complex standardization and deployment process.

However, we do have some engineering freedom in the link itself. Even if we do not want to modify the power control, there are other link-local mechanisms we can add or optimize: We could optimize the retransmission scheduling, taking advantage of the block loss correlation that we get after power control. We could use error correction coding. Or we could tweak the delay distribution by adding additional delays to selected packets.

In the remainder of this section, we investigate the simplest of these options, namely the third one.

A. Introducing additional delays

Assume that we have a discrete delay distribution X, $\text{P}(X = d_i) = p_i$, where $d_i < d_{i+1}$. It is typical, but not required, that also $p_i \geq p_{i+1}$.

We consider the following class of tweaks to X. For each packet that experiences a delay $X = d_i$, buffer the packet so that it gets an additional delay δ_i. This defines a new distribution $\tilde{X}$, $\text{P}(\tilde{X} = d_i + \delta_i) = p_i$ (or if it happens that $d_i + \delta_i = d_j + \delta_j$ for some $i \neq j$, the corresponding probabilities are added up). For an example of what X and $\tilde{X}$ can look like, see Figures 6 and 7.

What is the best choice for $\delta_i \geq 0$? One possible answer is given by the optimization problem

$$\begin{aligned}&\min_{\delta_i \geq 0} \mu(\tilde{X})\\ &\qquad \text{P}_{\text{TO}}(\tilde{X}) \leq \epsilon\end{aligned}$$

where $\epsilon > 0$ is a maximum allowed value for $\text{P}_{\text{TO}}(\tilde{X})$. This means that we want to push down our measure of TCP-unfriendlyness, while at the same time not adding more delay than necessary. We will see that after some simplifications, this is a quadratic optimization problem.

First, we require that $\text{P}_{\text{TO}}(\tilde{X})$ corresponds to a tail of the original distribution X. Let k be the smallest value such that $\sum_{i \geq k+2} p_i \leq \epsilon$. Let $c = d_{k+1} + \rho < d_{k+2}$, where $\rho \geq 0$ is a robustness margin. We impose the additional constraints $d_i + \delta_i \leq d_{k+1}$ for $i \leq k$, $\delta_i = 0$ for $i > k$, and $\text{RTO}(\tilde{X}) = c$. Then, for any δ_i satisfying these new constraints, we will have $\text{P}_{\text{TO}}(\tilde{X}) = \sum_{i \geq k+2} p_i \leq \epsilon$. We get the optimization problem

$$\begin{aligned}&\min_{\delta_1, \ldots, \delta_k} \mu(\tilde{X})\\ &\qquad \mu(\tilde{X}) + 4\sigma(\tilde{X}) = c\\ &\qquad 0 \leq \delta_i \leq d_{k+1} - d_i, \text{ for } i \leq k\end{aligned}$$

To write it in matrix form, let δ denote the vector $(\delta_1, \ldots, \delta_k)^T$, and similarly for p and d. Let $S = 16 \operatorname{diag} p - 17pp^T, b_i = 2p_i(16d_i + c - 17\mu), m_i = d_{k+1} - d_i$ and $\alpha = 16\sigma^2 - (c - \mu)^2$, where μ and σ denote the mean

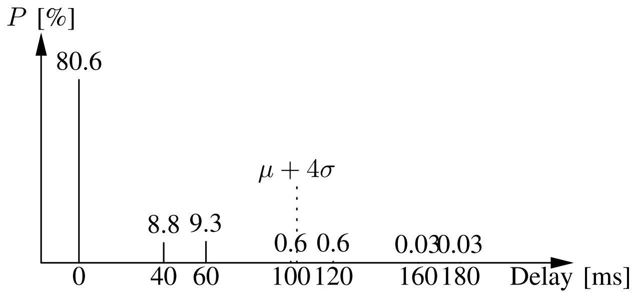

Fig. 6. Original delay distribution

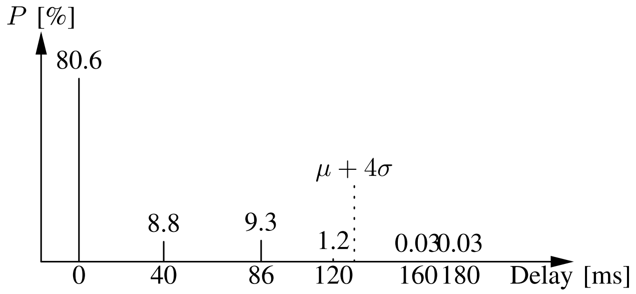

Fig. 7. Optimized delay distribution

and standard deviation of the original delay X. We can then rewrite the problem as

$$\min_{\delta_1,\ldots,\delta_k} p^T\delta$$
$$\delta^T S\delta + b^T\delta + \alpha = 0$$
$$0 \leq \delta_i \leq m_i$$

Remarks: Since the symmetric matrix S is typically indefinite, the problem is not convex. But it can be solved in exponential time $O(k^3 3^k)$, which has not been a problem thanks to the very limited size of k.

The rôle of ρ can be seen in Figure 7; it is the margin, on the delay axis, between the RTO $= \mu + 4\sigma$ and the delay value closest to the left (120 ms).

The typical solution is of the form $x = (0,\ldots,0,\delta_j,m_{j+1},\ldots,m_k)^T$. When the optimum has this form, it means that the cheapest way to increase the RTO, in terms of mean delay, is to increase the δ_i corresponding to the smallest p_i. Necessary and sufficient conditions for the optimum to be of this form has not yet been determined.

We aim at decreasing the distribution tail as measured by $\mathrm{P_{TO}}$, and pay a small price in mean delay. This is a different objective than an ordinary jitter buffer, which aims for small variance, and pays with a larger mean delay.

B. Numerical example

Now consider the delays d_i and probabilities p_i in Table I, and assume that packets are independently delayed according to the given probabilities. This distribution is also shown in Figure 6. Before tweaking the delays, we have $\mathrm{E}(X) \approx 10.6$ ms, and $\mathrm{P_{TO}}(X) \approx 0.68$ %.

With $\epsilon = 0.1\%$ and $\rho = 10$ ms, the above optimization procedure yields $k = 4$ and the optimal additional delay $\delta \approx (0, 0, 26, 20)^T$ ms. This modified distribution is shown in Figure 7. The mean additional delay is only 2.54 ms, which seems to be a small cost, if we compare it to the transmission delay for the packet, which is 40 ms, or the end-to-end delay which necessarily is even larger. We also achieve $\mathrm{P_{TO}} < \epsilon$, if fact, we actually get $\mathrm{P_{TO}} \approx 0.06\%$.

For the tweaked link, we have a slightly larger RTT (which in itself would decrease the throughput), and a significantly smaller $\mathrm{P_{TO}}$. The resulting throughput is 28.8 Kbyte/s, an improvement by 5% compared to the unmodified link, and only 1.4% below the ideal TCP throughput.

The important point is that a simple but carefully selected modification to the link-layer yields a modest but significant performance improvement.

V. Conclusions

In this contribution we have studied the effect retransmission in radio links has on packet properties. In particular, we have delineated the implications that it has on TCP. An input/output model has been suggested where the rôle of the scheduling mechanism is made explicit. The main contribution has been to show that a slight artificial increase of the delays of certain retransmitted packets may reduce the risk of spurious timeout in TCP and hence increase the throughput; in an example the increase was 5%. The artificial delay distribution is optimized off-line and applied on-line. The additional delay that is applied to a packet depends only on the retransmission delay experienced by that same packet, and this information is available locally at the link.

References

[1] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, pp. 314–329, 1988.

[2] 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.

[3] P. Sarolahti, M. Kojo, and K. Raatikainen, "F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, 2003.

[4] S. Cen, P. C. Cosman, and G. M. Voelker, "End-to-end differentiation of congestion and wireless losses," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 703–717, 2003.

[5] N. K. G. Samaraweera, "Non-congestion packet loss detection for TCP error recovery using wireless links," *IEE Proceedings-Communications*, vol. 146, no. 4, pp. 222–230, 1999.

[6] C. P. Fu and S. C. Liew, "TCP veno: TCP enhancement for transmission over wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216–228, 2003.

[7] H. Elaarag, "Improving TCP performance over mobile networks," *ACM Computing Surveys*, vol. 34, no. 3, pp. 357–374, 2002.

[8] R. G. Mukthar, S. V. Hanly, and L. L. H. Andrew, "Efficient Internet traffic delivery over wireless networks," *IEEE Communications Magazine*, pp. 46–53, December 2003.

[9] A. Dahlén and P. Ernström, "TCP over UMTS," in *Radiovetenskap och Kommunikation 02*, ser. RVK, 2002.

[10] N. Möller and K. H. Johansson, "Influence of power control and link-level retransmissions on wireless TCP," in *Quality of Future Internet Services*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2003, vol. 2811.

[11] 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.