Limitations of Fixed Timers for Wireless Links

George Xylomenos
xgeorge@aueb.gr
Mobile Multimedia Laboratory
Department of Informatics
Athens University of Economics and Business
Patision 76, Athens 104 34, Greece

***Abstract*— Traditional reliable link layer protocols set their fixed retransmission timers under the assumption that they operate in isolation over the link. Emerging wireless networks however allow multiple link layer sessions to dynamically share the link. To assess the impact of this development, we examine the performance of Web Browsing over a Selective Repeat protocol with fixed retransmission timers, showing that the optimal retransmission timer values depend on the level of contention. We therefore propose an adaptive Selective Repeat protocol that modifies its retransmission timers based on prevailing conditions. Our measurements show that this adaptive scheme provides excellent Web Browsing performance regardless of the level of contention, under two very different wireless error models.**

I. Introduction

Wireless networks are increasingly becoming an integral part of the Internet, especially in the role of access networks providing untethered connectivity to users. It is well known however that the error prone nature of wireless links degrades the performance of applications such as Web Browsing [10]. Reliable link layer protocols have been proposed as a way to hide the deficiencies of wireless links, thus improving the performance of the higher layer protocols and applications used on the Internet; the results reported in the literature show that reliable link layers do indeed offer dramatic performance improvements [11].

While traditional link layer protocols assume that they operate in isolation over the underlying link when setting their operating parameters, such as retransmission timers, emerging wireless networks allow multiple users and/or applications to dynamically share the link. This is most evident in the *Universal Mobile Telecommunications System* (UMTS) where a single physical channel is shared among independent link layer sessions employed by different users and/or applications. Since different applications have different requirements in terms of reliability and delay, when many applications share the same wireless link, different link layer protocols should expect to co-exist over it. While the sharing of a wireless link by independent link layer sessions also introduces fairness issues [11], in this paper we are only concerned with the interplay between link sharing and retransmission timer values and its effect on application performance.

The outline of this paper is as follows. In Sect. II we provide background on Internet protocol and application performance over wireless links and discuss related work. Section III describes our simulation setup for the performance evaluation that follows. In Sect. IV we describe the fixed Selective Repeat protocol used in this paper and discuss its performance with Web Browsing. Motivated by these results, in Sect. V we present an adaptive Selective Repeat protocol and evaluate its performance with Web Browsing against its fixed counterpart.

II. Background and Related Work

The heart of the Internet, the *Internet Protocol* (IP), offers an unreliable packet delivery service: packets may be lost, reordered or duplicated. Many real-time applications use the *User Datagram Protocol* (UDP) for direct access to this service, handling error, flow and congestion control themselves. Most other applications prefer delegating these tasks to the *Transport Control Protocol* (TCP) which offers a reliable byte stream service. TCP segments the application data stream into IP packets at the sender and reassembles it at the receiver. The receiver generates *acknowledgments* (ACKs) for segments received in sequence, returning duplicate ACKs for out of sequence segments. The sender retransmits the next unacknowledged segment either on receiving 3 duplicate ACKs or when a retransmission timer expires before an ACK is received.

Due to the high reliability of wired links, TCP assumes that all losses are due to congestion, thus after a loss it abruptly reduces its transmission rate to relieve congestion and then gradually increases it to probe the network. Unfortunately, losses due to wireless errors are also interpreted as congestion, causing TCP to dramatically reduce its transmission rate [10]. Many researchers have proposed TCP modifications to improve its performance over wireless links, but they all have two drawbacks: they require modifications to end hosts throughout the Internet and they can only retransmit lost data on an end-to-end basis.

Another approach is to employ a reliable link layer protocol over the wireless link so as to hide wireless errors from TCP. An early proposal customized to TCP *snoops* inside the packets of each TCP stream at the access point bridging the wired and wireless parts of the path and retransmits lost segments when duplicate ACKs arrive, hiding them from the sender to avoid end-to-end recovery [2]. Later work shows that the performance of TCP applications can be enhanced with standard reliable link layer protocols, without making the link layer TCP aware [11]. Avoiding TCP awareness has many advantages, such as compatibility with encrypted IP payloads which hide TCP headers from the link layer [5].

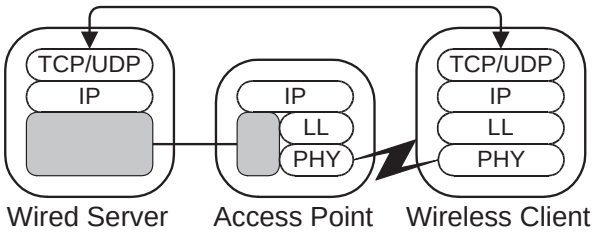

Fig. 1. Simulated network topology

An important issue with reliable link layer protocols is that not *all* Internet applications require their services. While TCP applications are well suited to them, delay sensitive UDP applications often prefer faster, albeit limited, error recovery. Reliable link layer sessions should therefore expect to co-exist with other link layers over the same wireless link. This causes the bandwidth available to the reliable link layer protocol, and therefore its effective *Round Trip Time* (RTT), to vary, leading to a problem when setting retransmission timers: they should be higher than the RTT, to prevent premature retransmissions, but not too high, to prevent the protocol from stalling until a timeout occurs. A TCP aware link layer sets its retransmission timers dynamically by mimicking TCP retransmissions [2], so as to retransmit lost packets before TCP, but this approach is inherently tied to TCP and not guaranteed to be the best one.

III. Simulation Setup

The performance results reported below are based on simulations with ns-2 [8], extended with additional error models, link layers and applications [9]. Each test was repeated 30 times with different random seeds. The results shown reflect average metric values from these 30 runs, as well as their 95% confidence intervals. The simulated topology is shown in Fig. 1: a Wired Server communicates with a Wireless Client via an Access Point. In all applications tested, the server was located at the wired end of the network and the client at the wireless end, hence the naming convention used. The wired link has a bandwidth of 10 Mbps and a propagation delay of 1 ms. Simulations using a 2 Mbps wired link with a propagation delay of 50 ms also support the conclusions reached in this paper.

The wireless link has a bandwidth of 64 Kbps, a propagation delay of 50 ms and uses a frame size of 250 bytes plus a header; these are typical characteristics for cellular links where bit interleaving inflates propagation delay. To avoid packet fragmentation, each application also uses 250 byte packets. Two error models were used for the wireless link. In the *Uniform* error model each frame may be independently lost with a probability of 1.5%, 2.5%, 5.4% or 9.8%. In the *Two State* error model the link can be either in a good state, with a bit error rate of 10^{-6}, or in a bad state, with a bit error rate of 10^{-2}. Both states have exponential durations, with the average duration of the good state being 10 s and the average duration of the bad state being 100 ms, 200 ms, 500 ms or 1000 ms. We have experimentally found that with these parameters the average *Frame Loss Rate* (FLR) of the Two State model is 1.5%, 2.5%, 5.4% or 9.8%, matching the FLRs used for the Uniform model. Note that the error processes in each link direction were identical but independent. To establish a performance baseline, we also show results with no errors.

To evaluate the Selective Repeat variants presented below, we used Web Browsing, the most popular application on the Internet [6], over TCP Reno with 10 ms granularity timers. In Web Browsing a client accesses *pages* containing text, links to other pages and embedded objects, stored on a server. The client-server interaction consists of *transactions*: the client requests a page from a server, the server returns the page which contains pointers to embedded objects, the client requests each embedded object, and the server returns them, completing the transaction. The next transaction begins when the client requests another page. The ns-2 HTTP module provides empirical distributions for request, page and embedded object sizes, as well as for the number of objects per page [6]. Only one transaction was in progress at any time with no pauses between transactions. The performance metric used was Web Browsing throughput, defined as the amount of all *application* data transferred from the server to the client divided by time taken. Client requests only influence throughput indirectly, by introducing delays. All results shown reflect the state at the end of the last completed transaction during the simulated period, which was 2000 s.

Contention is provided by a UDP real-time Media Distribution application. This application approximates a lecture where a speaker sends audio, and possibly video, to an audience including a wireless client. The speaker alternates between *talking* and *silent* states with exponential durations, averaging 1 s and 1.35 s, respectively [7], transmitting media only in the talking state. Packets are transmitted isochronously at a rate of 56 Kbps, consuming 87.5% of the available bandwidth in the talking state, but only 37.5% of the available bandwidth on average. As a result, the bandwidth available for Web Browsing is abruptly modified whenever Media Distribution changes state. It should be noted that no retransmissions are performed for the, delay sensitive, Media Distribution application. To be more exact, the Media Distribution data stream bypasses the reliable link layer protocol used by the Web Browsing data stream.

IV. Fixed Selective Repeat

In past research we have found the Selective Repeat protocol to offer excellent performance for TCP applications such as Web Browsing [11], without requiring TCP awareness. We have therefore decided to use it to study the interplay between link sharing and retransmission timers. In Selective Repeat, the sender transmits link layer frames in sequence within a transmission window of N frames, buffering them for possible retransmission. The receiver accepts frames within a reception window of N frames; if a frame is received in sequence it is delivered to the higher layer, the window slides upwards and an ACK is returned to the sender, confirming reception of all frames up to the one delivered. When the sender receives an ACK, it drops the buffered frames covered by it and also slides its window upwards.

When a frame arrives out of sequence at the receiver, it is buffered but not delivered, since the gap in the sequence indicates that some frames were lost; a *negative acknowledgment* (NACK) is returned for each missing frame to the sender, and the sender retransmits each NACKed frame. When missing frames arrive, the receiver delivers to the higher layer all frames that are now in sequence, slides its window upwards and returns an ACK covering all delivered frames. To reduce protocol overhead, in our implementation we delay returning an ACK for a short interval, trying to piggyback it into a data frame traveling in the reverse direction. If the interval expires, the ACK is sent as a separate frame. NACKs on the other hand are always sent immediately as separate frames.

If some ACKs and/or NACKs are lost, the sender may exhaust its transmission window, thus becoming unable to proceed. To prevent this, the sender starts a retransmission timer after sending each frame. If the timer expires before an ACK arrives for that frame, the frame is assumed lost and retransmitted. Many Selective Repeat variants exist, mostly differing on how NACKs are handled [3]. The variant used here allows each missing frame to be NACKed multiple times, a feature called *multireject*; we have also tested two simpler protocol variants, both of which support the conclusions reached in this paper.

We will now examine the performance of Web Browsing over Selective Repeat with fixed timers, in order to assess the effects of contention. Figure 2 shows the Web Browsing throughput achieved under the Uniform error model with a range of fixed timeout values from 0.9 s to 1.3 s, in 0.1 s increments. Throughput is maximized with the lowest timeout value, and it is progressively decreased as the timeouts are increased. The 9.8% performance gap, i.e. the difference between the best and worst options at a FER of 9.8%, is 13.4%. When contention is introduced however, the situation is completely reversed, as Fig. 3 shows: in this case lower timeout values perform worst, while higher timeouts lead to progressive improvement. In this case the 9.8% performance gap is 21.7%, but in the opposite direction than when no contention exists.

In the Two State error model, the situation without contention is not that clear. As Fig. 4 shows, it is hard to even distinguish between the various fixed timeout options, since the 9.8% performance gap is only 1.4%. However, when contention is introduced, Fig. 5 shows that throughput degrades as the timeouts are decreased, exactly as with the Uniform error model. Indeed, in this case the 9.8% performance gap is 40.9%, higher than with the Uniform error model.

These results indicate that under both wireless models, it is not possible to select a fixed retransmission timer value that will optimize overall Web Browsing performance: contention generally increases the optimal timeout value, due to the corresponding increase in the effective RTT. The results with contention show that a timeout value that provided excellent performance without contention, may exhibit terrible performance when contention is introduced. Therefore, with fixed timers the best we can do is to choose a timer that will provide a good compromise, rather than optimal performance.

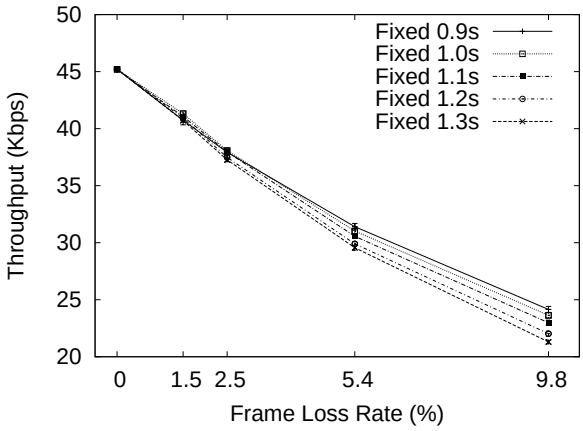

Fig. 2. Web Browsing Throughput without Contention (Uniform): Fixed Timeouts

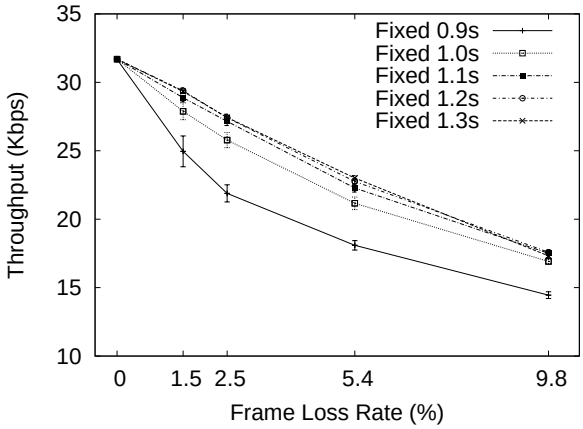

Fig. 3. Web Browsing Throughput with Contention (Uniform): Fixed Timeouts

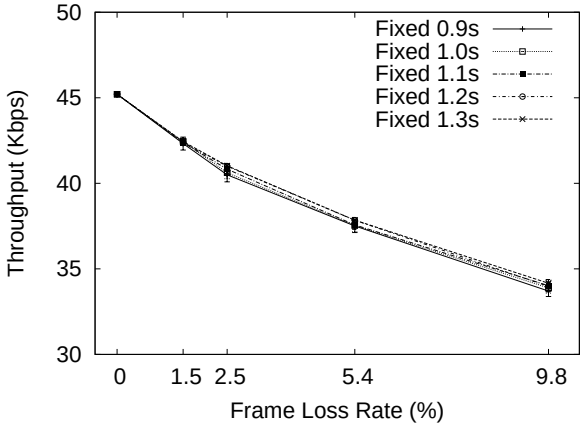

Fig. 4. Web Browsing Throughput without Contention (Two State): Fixed Timeouts

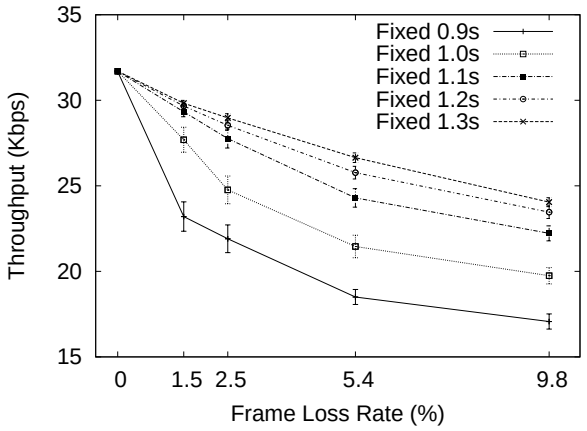

Fig. 5. Web Browsing Throughput with Contention (Two State): Fixed Timeouts

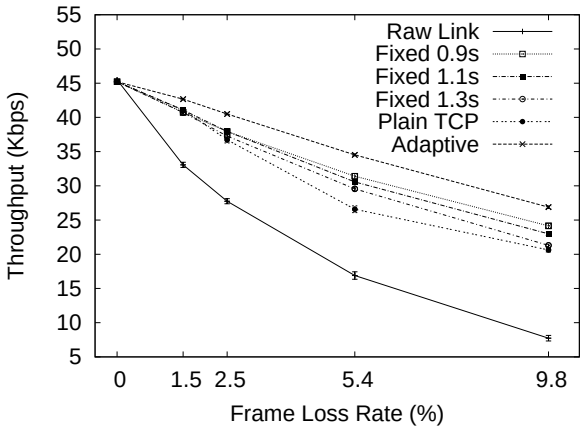

Fig. 6. Web Browsing Throughput without Contention (Uniform): Fixed vs. Adaptive Timeouts

V. Adaptive Selective Repeat

As shown in Sect. IV, contention for the link has a dramatic effect on performance: when competing sessions start and stop, the available bandwidth changes, influencing the effective RTT. It is thus desirable for a reliable link layer protocol to appropriately adapt its retransmission timers. To this end, we modified Selective Repeat to track the RTT in a manner similar to TCP [4]. For every packet transmitted or retransmitted, the sender notes its transmission time. When an ACK arrives for the packet, the difference between the current time and the transmission time provides an RTT $sample$. We use these samples to update smoothed estimates for the RTT, $srtt$, and its variance, $srttvar$, as follows:

$$srtt = 0.875 * srtt + 0.125 * sample \ . \quad (1)$$

$$srttvar = 0.75 * srttvar + 0.25 * (sample - srtt) \ . \quad (2)$$

As the effective RTT fluctuates, the estimators (1) and (2) follow its progress in a smoothed manner: they react to changes with a time lag and are not dramatically affected by sporadic extreme values. Note that the smoothing factors used are the same as those used by TCP, therefore these calculations can be performed very efficiently using integer arithmetic [4]. After updating the estimators, we calculate the new value to be used for the retransmission timers, $rtxto$, as follows:

$$\text{Uniform error model}: \ rtxto = 3 * srtt + 2 * srttvar \ . \quad (3)$$

$$\text{Two State error model}: \ rtxto = 4 * srtt + 0 * srttvar \ . \quad (4)$$

Note that both (3) and (4) differ from the formula used by TCP, which is $rtxto = 1 * srtt + 4 * srttvar$. We have experimentally found that these formulas perform very well for the corresponding wireless error models [12], in contrast to the plain TCP formula, whose performance will be discussed below.

Our adaptive timeout scheme calculates samples from *every* packet acknowledged, with three exceptions, meant to avoid inaccurate samples. First, NACKs do not provide samples, since they do not reflect reception of the NACKed frame. Second, when an ACK covers multiple frames, only the last frame acknowledged provides a sample, since the previous ones may have been received long ago. Third, when duplicate ACKs arrive, only the first ACK provides a sample; the following ones are ignored, since they do not offer additional information.

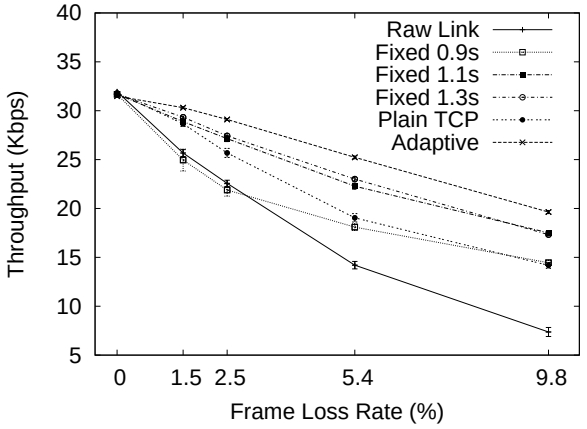

Fig. 7. Web Browsing Throughput with Contention (Uniform): Fixed vs. Adaptive Timeouts

We will now we examine the performance of Web Browsing over Selective Repeat with both adaptive and fixed timeouts. Figure 6 shows the Web Browsing throughput achieved under the Uniform error model with our *Adaptive* scheme, as well as with fixed timeout values of 0.9 s, 1.1 s and 1.3 s. We also show performance over the *Raw Link*, that is, without any link layer error recovery, and over *Plain TCP*, that is, when the standard TCP formula is used for timeout adaptation. Our adaptive scheme performs better than all fixed options: the 9.8% performance gap of our scheme over the fixed 1.1 s option, previously found to be a good compromise, is 17%, while plain TCP is worse than all fixed options.

When contention is introduced, Fig. 7 shows that our adaptive scheme again performs best; in this case the 9.8% performance gap over the fixed 1.1 s option is 12.1%. Interestingly, with the lowest fixed timeout, i.e. 0.9 s, which was the best option without contention, performance with contention can

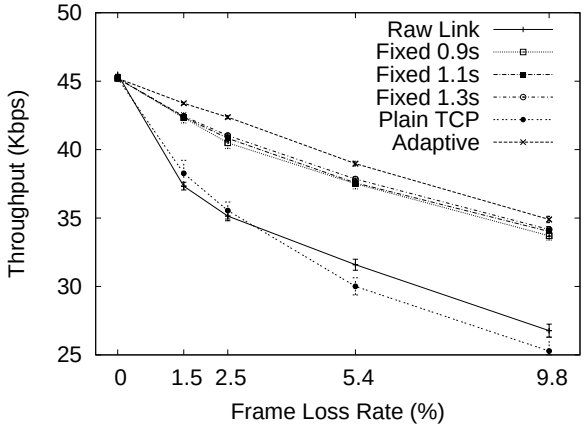

Fig. 8. Web Browsing Throughput without Contention (Two State): Fixed vs. Adaptive Timeouts

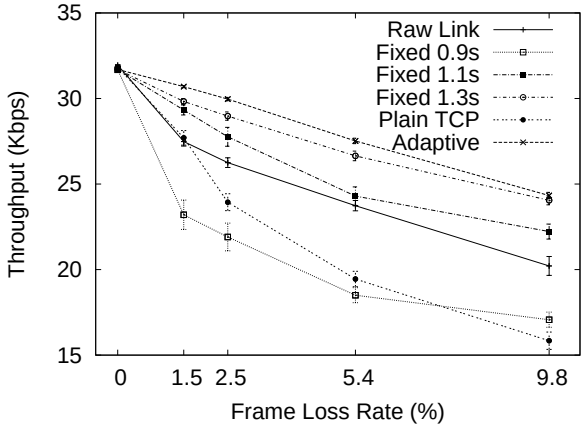

Fig. 9. Web Browsing Throughput with Contention (Two State): Fixed vs. Adaptive Timeouts

be worse than without any error control, indicating that timers expire early, leading to redundant retransmissions. The plain TCP formula also suffers from the same problem, therefore it only manages to beat the worst fixed timeout option.

The results with the Two State error model are very similar. Figure 8 shows that without contention our adaptive scheme outperforms all fixed schemes, with a 9.8% performance gap over the fixed 1.1 s option of 2.6%. When contention is introduced, Fig. 9 shows that our adaptive scheme again performs best, with a 9.8% performance gap over the fixed 1.1 s option of 9.5%. In this case, the use of a fixed timeout of 0.9 s, which was quite acceptable without contention, is always worse than no error control at all. Furthermore, it is clear that the plain TCP formula is completely inappropriate for the Two State error model, as it nearly always leads to lower performance than without any error recovery.

These results indicate that with Web Browsing our adaptive timeout scheme performs much better than the fixed scheme with the compromise timeout of 1.1 s, and, indeed, better than any of the fixed timeout options tested. More importantly, our adaptive scheme performs excellent without manual tuning regardless of the level of contention, in contrast to the fixed schemes where the choice of an optimal timeout value requires awareness of the, generally unknown, level of congestion over the link. Furthermore, our results show that the equation used by TCP is far from optimal, and may even be worse than performing no error control at all. Therefore, simply using the TCP policy is not enough.

We conclude this section with a brief sensitivity analysis of the parameters used for the adaptive scheme under each wireless error model. In previous work we have explored the parameter space for the coefficients of $srtt$ and $srttvar$, concluding that formulas (3) and (4) work best for the Uniform and Two State error models, respectively [12]. We have also compared our basic scheme, referred to as adaptive standard, against two variations: in the adaptive fast case, equations (1) and (2) are modified to $srtt = 0.75 * srtt + 0.25 * sample$ and $srttvar = 0.5 * srttvar + 0.5 * (sample - srtt)$, respectively, i.e. the estimators adapt faster to prevailing conditions; conversely, in the adaptive slow case, equations (1) and (2) are modified to $srtt = 0.9375 * srtt + 0.0625 * sample$ and $srttvar = 0.875 * srttvar + 0.125 * (sample - srtt)$, respectively, i.e. the estimators adapt slower. We omit these results for brevity, as all three variants of our adaptive scheme have only marginal performance differences, indicating that the scheme is relatively insensitive to the exact choice of the filter coefficients used.

VI. Conclusions and Future Work

We have discussed the problems faced by reliable link layer protocols when sharing a wireless link with competing link layer sessions, using Selective Repeat as an example. Our measurements of Web Browsing performance indicate that the optimal fixed retransmission timer values strongly depend on the level of contention for the link. We therefore proposed an adaptive Selective Repeat variant that dynamically sets its retransmission timers based on prevailing conditions, using a scheme similar to TCP. Our measurements indicate that this adaptive scheme outperforms its fixed counterparts regardless of the level of contention and frame loss rate, under two very different wireless error models. We have also found that, while our adaptive scheme is relatively insensitive to the filter coefficients used to smooth the average RTT and RTT variance estimators involved in the adaptive calculation of the timeout values, the actual coefficients used by TCP for this calculation are suboptimal for our link layer environment.

Our adaptive Selective Repeat approach is not the only way to overcome the issues introduced by contention at the link layer. The *Radio Link Control* (RLC) protocol used in UMTS networks in its Acknowledged Mode [1] does not use retransmission timeouts at the sender, relying solely on status information from the receiver to trigger retransmissions. Since both ACKs and NACKs may be lost however, either the RLC sender or the RLC receiver must periodically probe for or return status reports, respectively. We are currently implementing the UMTS RLC protocol in our simulator with the aim of comparing its performance against our adaptive Selective Repeat approach.

REFERENCES

[1] 3rd Generation Partnership Project (3GPP). Radio Link Control (RLC) protocol specification (Release 6). Technical Specification 25.322, V6.2.0, December 2004.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proc. of the ACM SIGCOMM '96*, pages 256–267, 1996.

[3] P. T. Brady. Evaluation of multireject, selective reject, and other protocol enhancements. *IEEE Trans. on Communications*, 35(6):659–666, June 1987.

[4] V. Jacobson. Congestion avoidance and control. In *Proc. of the ACM SIGCOMM '88*, pages 314–329, August 1998.

[5] S. Kent and R. Atkinson. IP encapsulating security payload (ESP). RFC 2406, 1998.

[6] B. A. Mah. An empirical model of HTTP network traffic. In *Proc. of the IEEE INFOCOM '97*, pages 592–600, 1997.

[7] S. Nanda, D. J. Goodman, and U. Timor. Performance of PRMA: a packet voice protocol for cellular systems. *IEEE Trans. on Vehicular Technology*, 40(3):584–598, 1991.

[8] UCB/LBNL/VINT. Network Simulator - ns (version 2). Available at http://www.isi.edu/nsnam.

[9] G. Xylomenos. Multi service link layers for ns-2. Available at http://www.mm.aueb.gr/~xgeorge/codes/codephen.htm.

[10] G. Xylomenos and G. C. Polyzos. Internet protocol performance over networks with wireless links. *IEEE Network*, 13(5):55–63, July/August 1999.

[11] G. Xylomenos and G. C. Polyzos. A multi-service link layer architecture for the wireless Internet. *International Journal of Communication Systems*, 17(6):553–574, 2004.

[12] G. Xylomenos and C. Tsilopoulos. Adaptive timeout policies for wireless links. In *Proc. of the International Conference on Advanced Information Networking and Applications*, volume 1, pages 497–502, 2006.