RLC protocol performance over TCP SACK

Michael Makidis and George Xylomenos
mmakidis05@cs.aueb.gr and xgeorge@aueb.gr
Mobile Multimedia Laboratory
Athens University of Economics and Business
Patision 76, Athens 104 34, Greece

***Abstract*—The Radio Link Control (RLC) protocol, used in Universal Mobile Telecommunication System (UMTS) networks, is one of the most advanced and complex link layer protocols. Among its notable features are the absence of retransmission timers, which makes it tolerant to contention for the link, and the ability to abandon persistently lost frames, which makes it suitable for reliable transport layers. In order to assess whether it makes sense to use RLC with non-UMTS wireless links, especially in the face of the enhanced error recovery offered by TCP with the Selective Acknowledgment (SACK) option, we implemented it in the ns-2 simulator and measured the throughput achieved by File Transfer and Web Browsing over RLC, either with or without contention from a Media Distribution application. While we found that RLC adapts well to the available bandwidth, providing considerable performance gains with random losses, with bursty losses RLC hardly improved upon TCP SACK.**

I. Introduction

As wireless networks become commonplace, it has become evident that TCP performance over wireless links leaves a lot to be desired [1]. The most direct method for improving TCP performance is to use a reliable link layer protocol to hide wireless losses from TCP, and this is the option adopted by the *3rd Generation Partnership Project* (3GPP) in its *Universal Mobile Telecommunications System* (UMTS) specifications. The UMTS data link layer protocol, the *Radio Link Control* (RLC) [2] protocol, incorporates many advanced features. For example, since in UMTS networks multiple link layer sessions share the wireless medium, making the effective round trip delay unpredictable, the RLC sender does not employ retransmission timers to detect packet losses; instead, it relies only on explicit receiver status reports. Furthermore, persistently lost frames are dropped, so as to prevent the link layer from stalling, assuming that reliable end-to-end protocols, such as TCP, will deal with residual errors.

While RLC was designed for UMTS networks, these features are potentially useful for other shared wireless links, such as wireless local area networks. The question that we attempt to answer in this paper is whether introducing the complexity of RLC in a non-UMTS wireless link is justified when a modern TCP implementation is employed, in particular, TCP with the *Selective Acknowledgment* (SACK) option and fine granularity timers. While TCP SACK was designed to better handle congestion losses [3], its improved loss recovery can also improve (but not optimize) TCP performance over wireless links [4]; fine granularity timers, used in most current TCP implementations [5], make TCP react faster to losses.

Our work extends most previous RLC performance studies which employ TCP Reno with coarse granularity timers [6], [7], [8]. Furthermore, while most previous studies focus on File Transfers [9], we also examine Web Browsing. Finally, while previous studies examine TCP performance in isolation, we introduce contention for the link in the form of a Media Distribution application, in order to assess whether RLC can indeed adapt to fluctuations in the available bandwidth.

The outline of the rest of this paper is as follows. In Section II we provide an overview of the RLC protocol and its options, while in Section III we present the simulation setup for the performance evaluation that follows. In Section IV we examine the performance of Web Browsing and File Transfer over RLC with and without contention for the link from a Media Distribution application, using both random and bursty wireless loss models. We summarize our findings in Section V.

II. RLC Protocol Overview

Since the use of RLC is mandatory for all UMTS link layer sessions, three modes are supported. In *Transparent Mode* (RLC/TM) the protocol passes all frames unmodified to the lower layer; this mode is used for voice calls. In *Unacknowledged Mode* (RLC/UM) the protocol optionally performs duplicate avoidance and reordering; this mode is used by applications requiring low delay, such as media streaming. Finally, in *Acknowledged Mode* (RLC/AM) the protocol performs error detection and recovery; this mode is used by applications requiring a reliable channel, such as TCP based ones. In this paper we focus exclusively on RLC/AM.

RLC/AM is essentially a sliding window *Automatic Repeat reQuest* (ARQ) protocol. The sender is passed variable size packets from the higher layer, called *Service Data Units* (SDUs), it segments and/or packs the SDUs into a sequence of fixed size frames, called *Protocol Data Units* (PDUs), and transmits these frames to the receiver. The receiver uses the sequence of incoming PDUs to reassemble the original SDUs and deliver them to the higher layer. The sender (receiver) may only transmit (receive) PDUs within a sender (receiver) window; the windows slide upwards as PDUs are received in sequence and acknowledged.

The sender labels each new *Acknowledged Mode Data* (AMD) PDU with a *Sequence Number* (SN) before transmission and buffers the PDU for possible retransmission. The receiver detects losses when out-of-sequence AMD PDUs arrive and informs the sender about received and missing AMD

WIOPT 2008, 1st – 3rd Apr 2008, Berlin, Germany.
 ISBN 978-963-9799-18-9
DOI 10.4108/ICST.WIOPT2008.3188

PDUs by returning *Status* PDUs. A Status PDU consists of one or more *Super Fields* (SUFIs). All RLC/AM implementations must support at least the ACK SUFI which acknowledges all SNs up to the one indicated, while other SUFIs are optional. A novel feature of RLC/AM is the BITMAP SUFI, indicating the status of the receiver window after the latest AMD PDU received in-sequence: each bit in the map indicates whether the corresponding PDU has been received or not.

Status PDUs may be piggybacked in the padding of the (fixed size) AMD PDUs, or sent as independent frames. Unlike in other ARQ protocols, in RLC/AM Status PDUs are *not* automatically returned after every frame received: they are either requested by the sender or returned by the receiver based on various triggers. Therefore, RLC/AM cannot rely on retransmission timers to detect lost PDUs, as the Status PDU acknowledging a correctly received frame may take some time to be returned. This absence of retransmission timers makes RLC/AM robust to contention for the link, since it decouples the protocol from the round trip delay of the link. However, it also means that some other mechanism is needed to ensure that Status PDUs are eventually returned to the sender, so as to avoid the possibility of deadlocks [8].

The sender can request a Status PDU to be returned by setting a polling bit in the AMD PDU header. Polling may be triggered by a number of events: (a) when the last PDU in the transmission (or retransmission) buffer is transmitted, (b) after every x PDUs (or SDUs) transmitted, (c) when the sender's window usage is higher than a configured limit, (d) when a periodic timer expires, and, (e) when a timer set after sending a poll expires without receiving acknowledgments for the AMD PDUs outstanding when that poll was sent.

In addition to explicit polls from the sender, the receiver may also return a Status PDU if any of the following events occur: (a) gaps are detected in the received AMD PDU sequence, (b) a periodic timer expires, and, (c) a request arrives from the lower layers. Due to the numerous options for triggering polls and status reports, RLC/AM defines two optional functions to limit the number of the Status PDUs returned. If *Poll Prohibit* is used, the sender starts a timer after a poll and defers further polls until the timer expires. If *Status Prohibit* is used, the receiver starts a timer after a Status PDU and defers further Status PDUs until the timer expires.

RLC/AM also departs from other ARQ protocols in that it abandons persistently lost PDUs, thus trading off reliability for delay. The persistent loss of a PDU causes the entire SDU of which that PDU is a part of to be discarded. When the *SDU Discard* function is triggered, the sender notifies the receiver to advance its window by transmitting a *Move Reception Window* (MRW) SUFI in a Status PDU, the receiver advances its window and acknowledges by returning an MRW_ACK SUFI, and the sender finally advances its window. The sender can be configured to use one of three modes: (a) discard unacknowledged PDUs when a period of time elapses, (b) discard unacknowledged PDUs after a number of unsuccessful transmissions, or (c) reset the peer RLC/AM entities after a number of unsuccessful transmissions.

III. Simulation Setup

The performance results reported below are based on simulations with ns-2 [10], extended with additional error models and link layer protocols. Our RLC/AM implementation conforms to the relevant 3GPP specification [2], except for a few features that are irrelevant for non-UMTS links; see [11] for details. Each test was repeated 30 times with different random seeds and the results shown reflect average metrics from these 30 runs, as well as their 99% 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 at the wired end of the network and the client at the wireless end. For the wired link we used a bandwidth of 2 Mbps and a propagation delay of 50 ms, simulating a multi-hop wide area path.

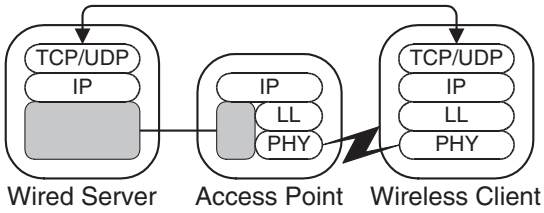


Fig. 1. Simulated network topology.

Even though we are not concerned with UMTS links, the parameters of the simulated wireless links are similar to those of UMTS ones: the bandwidth is 64 Kbps, just sufficient for the applications under study, the propagation delay is 50 ms, incorporating interleaving, framing and propagation delays, and the frame (PDU) size is 250 bytes. To avoid fragmentation, the application packet (SDU) size was also set to 250 bytes. 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; with these parameters the average *Frame Loss Rate* (FLR) of the Two State model turns out to be 1.5%, 2.5%, 5.4% or 9.8%, matching the FLRs of the Uniform model. The error processes in each direction were identical but independent.

Since File Transfer performance cannot characterize the performance of interactive applications [1], we measured both File Transfer and Web Browsing performance over TCP SACK with 10 ms granularity timers, as used in most current TCP implementations [5]. In File Transfer, a file was transmitted from the wired server to the wireless client. As users do not make infinite transfers, we used a file size of 10 Mbytes. The ns-2 File Transfer module sends data as fast as possible, with TCP handling flow, error and congestion control. We measured File Transfer throughput, defined as the amount of application data transferred divided by total time.

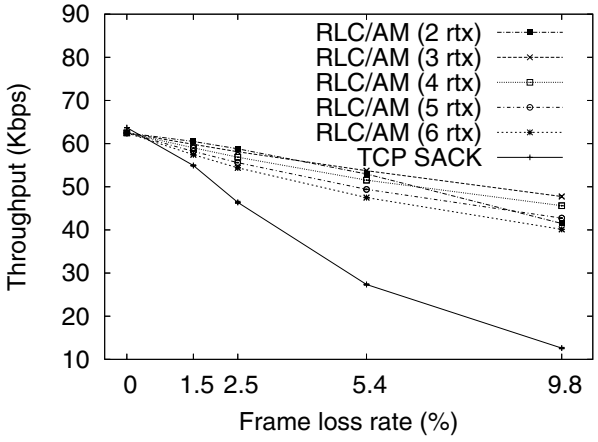
TABLE I
RLC/AM CONFIGURATION PARAMETERS.

Parameter	Uniform	Two State
Window size	128 frames	
Piggybacked Status	No	
Poll triggers	Poll Timer, Window Based	Poll Timer, Window Based, Every Last PDU in Retr. Buffer
Poll Timer timeout	200 ms	
Window threshold	70%	80%
Poll Prohibit	Yes (100 ms)	
Status Report triggers	Missing PDU, Timer based	
Status Report timeout	400 ms	500 ms
Status Prohibit	No	Yes (90 ms)
SDU Discard mode	Discard after x transmissions	
Maximum (re)transmissions	2 to 6	

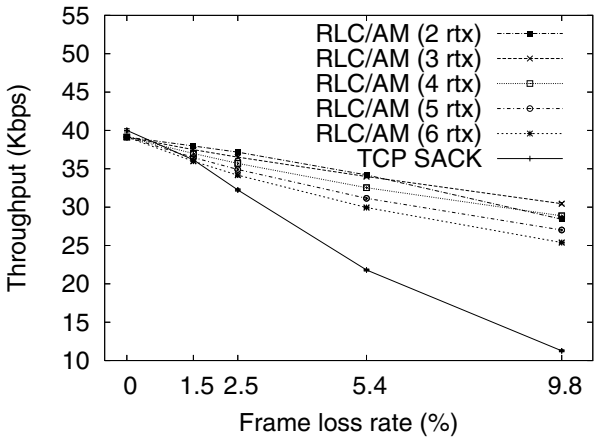
In Web Browsing a client accesses *pages* containing text, links and embedded objects, stored on a server. The client-server interaction consists of *transactions*: the client requests a page from a server, the server returns the page which contains pointers to embedded objects, the client requests each embedded object, and the server returns them, thus completing the transaction. The ns-2 Web Browsing module provides empirical distributions for request, page and embedded object sizes, as well as for the number of objects per page. Only one transaction was in progress at any time with no pauses between transactions. We measured Web Browsing throughput, defined as the amount of application data transferred from the server to the client divided by total time. The simulation period for Web Browsing was 2000 s.

Since RLC/AM does not use retransmission timers, in principle it can cope with contention for the link. We introduced contention in the form of a UDP based Media Distribution application, approximating a lecture where a speaker sends video to a wireless listener. The speaker alternates between *talking* and *silent* states with exponential durations, averaging 1 s and 1.35 s, respectively, transmitting media only in the talking state; no retransmissions are performed. Packets are transmitted isochronously at a rate of 56 Kbps, consuming 87.5% of the available bandwidth in the talking state.

Previous studies have found that the SDU discard policy [6], the RLC window size [7], the poll trigger policy [8] and the status prohibit policy [9] have an impact on TCP performance over RLC/AM with UMTS links. We fine tuned the RLC/AM parameters for the links under study based on an extensive simulation study [12]. The RLC window size was 128 frames, with Status PDUs transmitted as separate frames (not piggybacked). Polls were triggered at the sender whenever the transmission window occupancy reached 70% for Uniform links and 80% for Two State links, or when a 200 ms poll timer expired; on Two State links polls were also triggered whenever the retransmission buffer was exhausted; after triggering a poll, further polls were deferred for 100 ms. Status PDUs were also triggered at the receiver whenever a missing PDU was detected, as well as every 400 ms for Uniform links and every 500 ms for Two State links; after triggering a Status PDU, further Status PDUs were deferred for 90 ms on Two State links. In order to assess the performance impact of the limited persistence feature of RLC/AM on TCP applications, we discarded PDUs (and therefore SDUs) after 2 to 6 (re)transmissions, including the original transmission. Table I summarizes all the RLC/AM parameters.



(a) No Contention

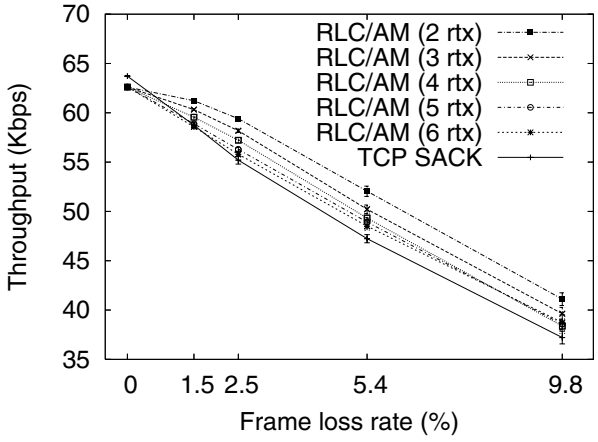


(b) Contention

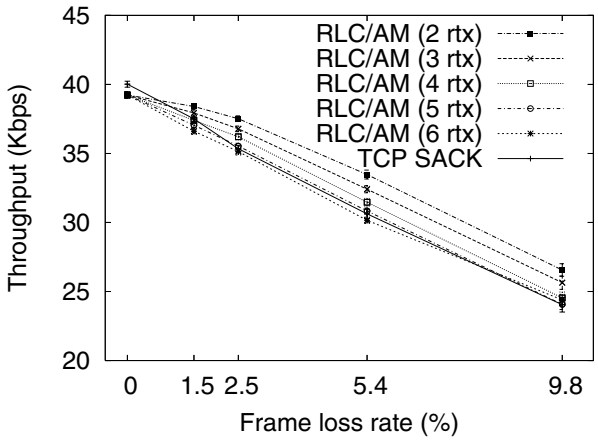
Fig. 2. File Transfer throughput (Uniform error model)

IV. PERFORMANCE EVALUATION

Starting with File Transfer throughput, Fig. 2(a) shows that with the Uniform error model and no contention, RLC/AM provides a considerable improvement over TCP SACK. The slight performance lag of RLC/AM at zero FLR, also evident in later figures, is due to its additional overhead, as Status PDUs were not piggybacked on AMD PDUs. The retransmission limit turns out to be a critical parameter: as it is increased from 3 to 6 retransmissions, performance gradually drops. The best performance is provided with 2 retransmissions at low FLRs, while at higher FLRs 3 retransmissions are preferable. When contention is introduced, Fig. 2(b) shows that the relative performance of TCP SACK and RLC/AM with varying numbers of retransmissions remains nearly the same.

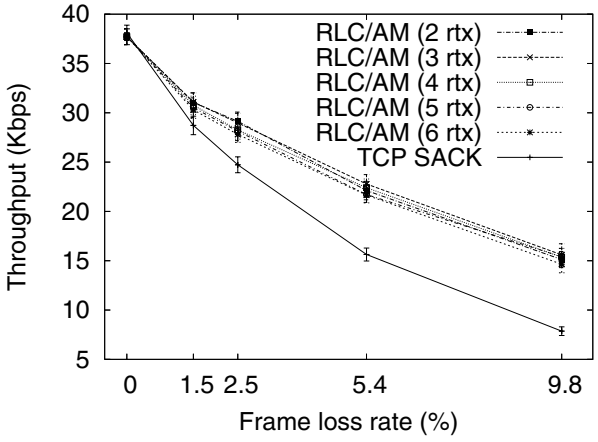


(a) No Contention

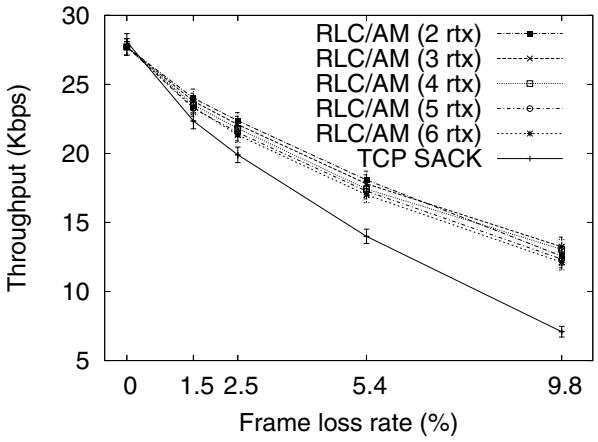


(b) Contention

Fig. 3. File Transfer throughput (Two State error model)



(a) No Contention



(b) Contention

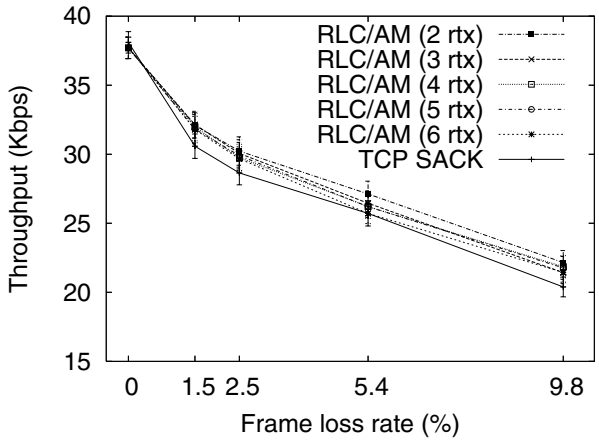
Fig. 4. Web Browsing throughput (Uniform error model)

Since the performance gap between 2 and 3 retransmissions is small at lower FLRs, 3 retransmissions seem to be a better choice overall. At an FLR of 5.4%, RLC/AM with 3 retransmissions increases File Transfer throughput by 96.51% without contention and by 55.88% with contention.

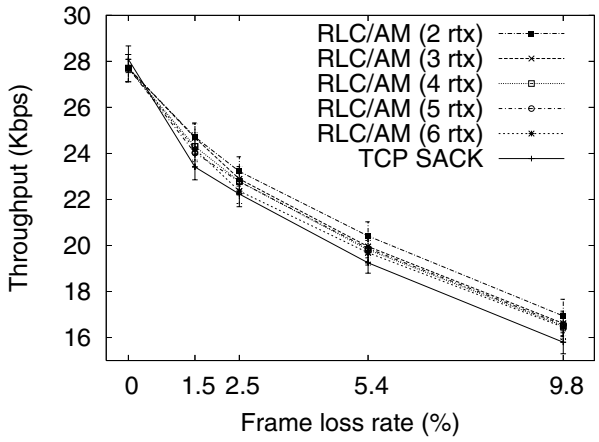
With the Two State error model however, the situation is quite different. Fig. 3(a) shows that without contention for the link, RLC/AM only provides a small performance advantage over TCP SACK. This is also the case when contention is introduced as Fig. 3(b) shows; indeed, with a high retransmission limit RLC/AM can even reduce performance. As with the Uniform error model, higher numbers of retransmissions provide progressively lower performance, and the relative performance of TCP SACK and RLC/AM is nearly the same regardless of the level of contention. A limit of 2 retransmissions provides the best performance either with or without contention, in this case regardless of the FLR. At an FLR of 5.4%, RLC/AM with 2 retransmissions increases File Transfer throughput by 10.18% without contention and by 9.30% with contention.

Turning to Web Browsing throughput, Fig. 4(a) shows that with the Uniform error model and without contention, RLC/AM again provides a considerable improvement over TCP SACK. While the retransmission limit has a smaller effect, 2 retransmissions still provide the best performance at lower FLRs and 3 retransmissions at higher FLRs; more retransmissions lead to decreased performance. The situation is nearly the same when contention is introduced, as Fig. 4(b) shows. In both cases, 3 retransmissions provide the best overall performance. At an FLR of 5.4%, RLC/AM with 3 retransmissions increases Web Browsing throughput by 45.73% without contention and by 27.21% with contention.

With the Two State error model, the situation is again quite different, as in the File Transfer case. Fig. 5(a) shows that without contention for the link, the advantage of RLC/AM over TCP SACK is minor, and this is also the case when contention is introduced as Fig. 5(b) shows. A limit of 2 retransmissions always provides the best performance; more retransmissions provide lower performance, although the performance is never worse than without RLC/AM. At an FLR of 5.4%, RLC/AM with 2 retransmissions increases Web Browsing throughput by 5.47% without contention and by 6.03% with contention.

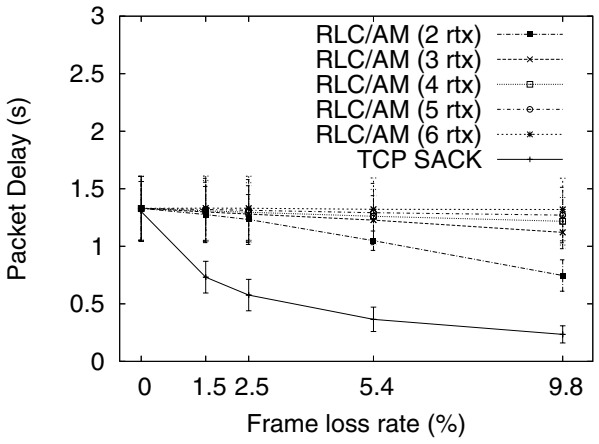


(a) No Contention

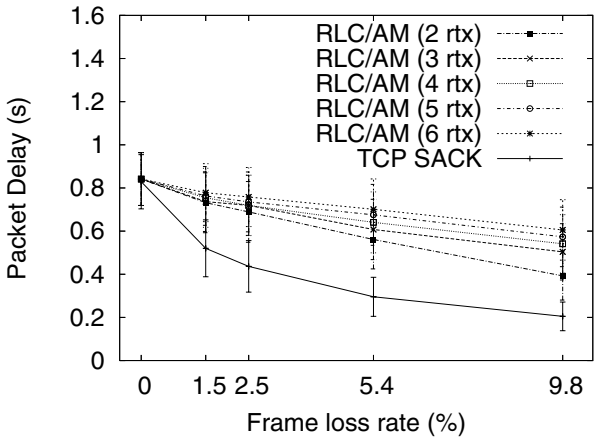


(b) Contention

Fig. 5. Web Browsing throughput (Two State error model)



(a) Contention from File Transfer

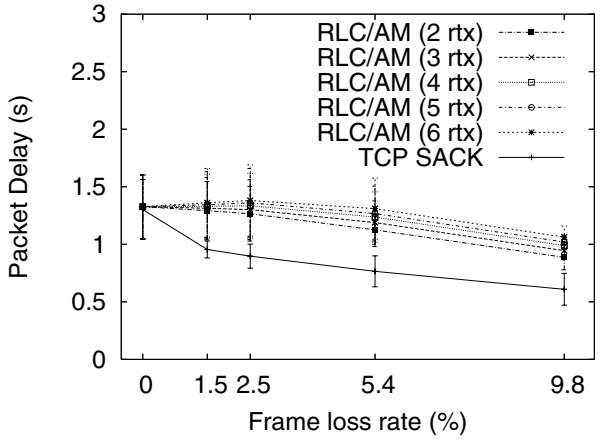


(b) Contention from Web Browsing

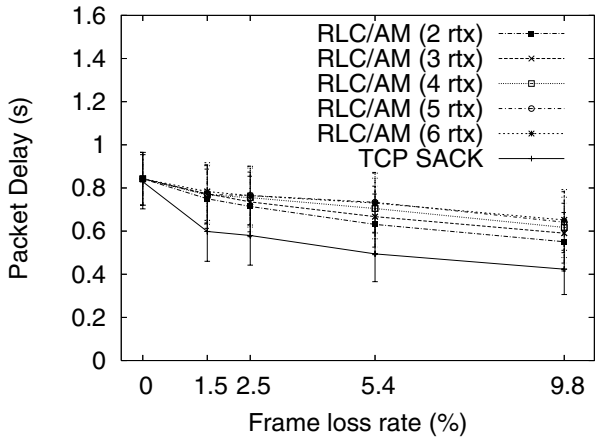
Fig. 6. Media Distribution delay (Uniform error model).

Since RLC/AM abandons persistently lost frames, thus preventing retransmission conflicts between the link and transport layers, it should be more friendly to applications contending for the link. Figure 6(a) and Figure 6(b) show Media Distribution packet delay when contending against File Transfer and Web Browsing, respectively, with the Uniform error model while Figure 7(a) and Figure 7(b) show the same metric when Media Distribution contends against File Transfer and Web Browsing, respectively, with the Two State error model. While the introduction of RLC/AM does lead to increased contention for the link, and therefore increased delays for Media Distribution packets, limiting the number of retransmissions per frame leads to bounded delays. And even though more retransmissions mean higher delays, it must be noted that at the optimal retransmission limits, that is, 3 retransmissions for Uniform and 2 retransmissions for Two State links, the Media Distribution delay does not increase at higher FLRs, despite the need for more and more retransmissions.

These results can be summarized in five observations. First, RLC/AM provides a higher performance improvement over TCP SACK with the Uniform error model than with the Two State error model, regardless of the application in use. The reason is that while RLC/AM can recover equally well from random and bursty losses due to its BITMAP SUFIs, TCP loss recovery is actually more efficient with bursty rather than with random losses, especially with the SACK option that can recover from multiple losses within a single round trip time [3]. Second, RLC/AM is more effective with File Transfer than with Web Browsing, providing roughly twice the performance gain. This is due to the short transfers typical of Web Browsing: when the last packet of a transfer is lost, RLC/AM will not detect the loss until a Status PDU is asked for or returned due to a timer expiration, and these timers are conservative to avoid overloading the link with Status PDUs. Third, while the optimal RLC/AM retransmission limit depends on the error model, it is generally low, with more retransmissions leading to progressively lower throughput with both error models. This is because the end-to-end delay is not much higher than the wireless link delay and TCP SACK uses fine granularity timers, therefore if RLC/AM does not recover from a loss with a few retransmissions, TCP SACK will also timeout and retransmit the same data, leading to a



(a) Contention from File Transfer



(b) Contention from Web Browsing

Fig. 7. Media Distribution delay (Two State error model).

waste of bandwidth. Fourth, RLC/AM shows the same overall behavior either with or without contention, reaching its peak performance with the same retransmission limit for each error model, regardless of the application, meaning that it can easily adapt to the bandwidth available on the link. Fifth, even though the introduction of RLC/AM leads to higher packet delays for the contending applications, since more retransmissions take place over the link, the limited retransmission feature of RLC/AM places an upper bound on these delays without sacrificing TCP application performance.

V. Conclusions

We presented the UMTS RLC/AM protocol and evaluated its performance in conjunction with TCP SACK and different applications, wireless error models and contention levels. We have found that RLC/AM (a) provides higher benefits with random rather than with bursty errors, (b) works better with bulk transfer rather than with interactive applications, (c) operates best with a low frame retransmission limit, (d) offers roughly the same performance gains either with or without contention for the link, and, (e) only leads to a minor increase in the delay of contending applications. We therefore conclude that, while RLC/AM is an excellent complement to TCP SACK for links with random losses, its high complexity is scarcely justified by its performance in links with bursty losses, where TCP SACK provides virtually the same performance.

References

[1] G. Xylomenos and G. C. Polyzos, "A multi-service link layer architecture for the wireless Internet," *International Journal of Communication Systems*, vol. 17, no. 6, pp. 553–574, 2004.

[2] 3rd Generation Partnership Project (3GPP), "Radio Link Control (RLC) protocol specification (Release 7)," Technical Specification 25.322, V7.0.0, March 2006.

[3] K. Fall and S. Floyd, "Simulation based comparisons of Tahoe, Reno and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, pp. 5–21, July 1996.

[4] G. Xylomenos and G. V. Kokkinakis, "Multiple layer error control over wireless links," in *Proc. of the European Wireless Conference (EW)*, 2007.

[5] S. Rewaskar, J. Kaur, and F. D. Smith, "Performance study of loss detection/recovery in real-world TCP implementations," in *Proc. of the IEEE International Conference on Network Protocols (ICNP)*, 2007.

[6] F. Lefevre and G. Vivier, "Optimizing UMTS link layer parameters for a TCP connection," in *Proc. of the IEEE Vehicular Technology Conference (VTC) Spring*, vol. 4, 2001, pp. 2318–2322.

[7] R. Bestak, R. Godlewski, and P. Martins, "RLC buffer occupancy when using a TCP connection over UMTS," in *Proc. of the IEEE Personal, Indoor and Mobile Radio Communications (PIMRC) Conference*, vol. 3, 2002, pp. 1161–1165.

[8] M. Rossi, L. Scaranari, and M. Zorzi, "On the UMTS RLC parameters setting and their impact on higher layers performance," in *Proc. of the IEEE Vehicular Technology Conference (VTC) Fall*, vol. 3, 2003, pp. 1827–1832.

[9] J. Alcaraz, F. Cerdan, and J. Garcia-Haro, "Optimizing TCP and RLC interaction in the UMTS radio access network," *IEEE Network*, vol. 20, no. 2, pp. 56–64, 2006.

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

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

[12] M. Makidis, "Implementing and evaluating the RLC/AM protocol of the 3GPP specification," Master's thesis, Dept. of Informatics, Athens University of Economics and Business, Available at http://mm.aueb.gr/archive.html, February 2007.