Wireless link layer enhancements for TCP and UDP applications

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

***Abstract*— Internet application performance over wireless links is disappointing due to wireless impairments that adversely affect higher layers. This paper focuses on link layer enhancement mechanisms that hide wireless errors from the rest of the Internet. We simulated file transfer and WWW browsing over TCP and continuous media distribution over UDP, in conjunction with various link layer schemes. Our results reveal that WWW browsing has substantially different behavior than file transfer, that existing TCP enhancement schemes have limited applicability and that UDP applications are best served by schemes inappropriate for TCP. Therefore, multiple link layer solutions are needed to optimize the performance of diverse applications.**

I. Introduction

The Internet has always been quick to adopt new communications technologies, largely due to IP which offers a common interface to higher layers, regardless of the underlying link. The growth of the Internet is only paralleled by the growth in wireless communications. *Cellular Telephony* is constantly evolving towards higher bit rates, while *Wireless Local Area Networks* (WLANs) have become inexpensive and interoperable. The popularity of such wireless systems makes their integration into the Internet very important. Even though satellites have long been a part of the Internet, higher layers still make assumptions about link performance that cannot be met by wireless links. Thus, although supporting IP over these links is easy, the resulting performance is poor [1]. As the Internet evolves towards *Quality of Service* (QoS) provision, improving Internet application performance over wireless links becomes critical.

This paper evaluates a range of link layer mechanisms aiming to improve Internet application performance over wireless links. In Section II we review related work. Section III describes our simulation setup. We then discuss the performance of three applications, using various link layers: Section IV covers file transfer, Section V covers World Wide Web browsing and Section VI covers continuous media distribution. Our results reveal that WWW browsing differs substantially from file transfer, that TCP aware enhancement approaches are limited in practice and that UDP applications favor fundamentally different enhancement schemes than TCP ones. Section VII presents our conclusions.

II. Background and Related Work

IP offers an unreliable packet delivery service. IP packets may be lost, reordered or duplicated. Applications can use UDP for direct access to this service when they expect the network to be reliable enough for their needs. For example, UDP is employed for file sharing via NFS over wired LANs. Delay sensitive applications may also use UDP, adding customized error recovery mechanisms. For example, real-time conferencing applications may encode their data to tolerate losses without resorting to retransmissions.

Most applications are not error tolerant, thus they employ TCP which offers a reliable byte stream service. TCP segments and reassembles application data into IP packets. The receiver generates cumulative *acknowledgments* (ACKs) for segments received in sequence, with duplicate ACKs for out of sequence segments. Since IP may reorder packets, the sender retransmits the next unacknowledged segment only after receiving multiple (usually 3) duplicate ACKs. The sender dynamically tracks the round trip delay of the connection, so that if a segment is not acknowledged on time, it is retransmitted. TCP assumes that all losses are due to congestion, thus after a loss it reduces its transmission rate so as to allow routers to drain their queues, and then gradually increases this rate to probe the network [2].

This paper focuses on the, widely available, Cellular and WLAN systems. These links are slower and less reliable than wired ones, with lower delays than satellites. Current Cellular systems support low bit rates in the wide area while WLANs provide higher speeds within a few hundred feet. Wireless errors causes UDP application performance to degrade or become unacceptable, due to UDP's lack of recovery mechanisms. On the other hand, TCP continuously reduces its transmission rate under frequent errors, so as to avoid what it (falsely) assumes to be congestion, dramatically reducing throughput. Higher delay paths are more sensitive, since end-to-end recovery is slower on such paths. With multiple wireless links on the path throughput is reduced in a multiplicative manner [1].

The performance of UDP applications over wireless links has not been extensively studied in the past, mainly due to their diversity. UDP applications were also perceived as LAN oriented, a situation challenged by UDP-based multimedia streaming on the Internet. Considerable work has been devoted to TCP however. Most TCP enhancement schemes try to avoid triggering

congestion recovery due to wireless errors. General purpose TCP enhancements such as *Eifel* [3] and *Selective Acknowledgments* [4] improve TCP performance by reducing the *number* of redundant TCP retransmissions, without reducing their *delay* though.

A wireless specific approach is to *split* TCP connections into one connection over the wireless link and another one over the wired part of the path, bridged by an agent [5]. Recovery is performed locally over the wireless link. This violates the end-to-end semantics of TCP and is incompatible with IP security which encrypts TCP headers [6]. TCP is also not designed for fast error recovery, so it is inefficient even over a single link. Other approaches maintain TCP semantics by *freezing* TCP state whenever persistent errors are detected [7], [8]. As long as error conditions persist, TCP refrains from invoking its congestion control mechanisms. Wireless and congestion losses are differentiated either via explicit loss notifications [7] or via explicit congestion notifications [8]. Thus performance is not unnecessarily degraded, but error recovery remains end-to-end and TCP has to be modified or amended with extra software.

The alternative is local error recovery at the link layer, as in the *Radio Link Protocols* (RLPs) of Cellular systems [9], [10]. RLPs apply a single scheme which may be inappropriate for some traffic. For example, retransmissions unduly delay real-time traffic, so they should be bypassed in that case. RLPs may also interfere with TCP [11], leading to conflicting retransmissions between the link and transport layers. Therefore, link layer retransmissions should only be used when they are much faster than end-to-end ones.

One method of jointly optimizing recovery between layers is to exploit transport layer information at the link layer. By *snooping* inside *each* TCP stream at the wireless base station we can transparently retransmit lost segments when duplicate ACKs arrive, hiding the duplicates from the sender to avoid end-to-end recovery, thus reducing control overhead and avoiding cross layer interactions [12]. This approach is incompatible with IP security as it employs TCP header information and it only works in the direction from the wired Internet towards a wireless host due to its reliance on TCP ACKs. In the reverse direction, ACKs are returned late and may even signify congestion losses [1].

Link layer approaches have the advantage over higher layer ones of providing a solution that can be locally deployed, without requiring changes to higher layers. They are thus transparent to the rest of the Internet, faster to recover than transport layer solutions and able to exploit lower layer information to optimize error recovery. This paper concentrates on the issues of which schemes can optimize the performance of diverse applications and whether a single link layer scheme is sufficient for all needs.

III. Simulation Setup

To study the interactions between link layer schemes, wireless error models, transport protocols and applications, we performed simulations using ns-2 [13]. In addition to its realistic TCP modules, the ns-2 source code is freely available. We implemented additional wireless error modules, link layer schemes and application models [14]. To compensate for statistical fluctuations we repeated each test 30 times, using the random seeds embedded in ns-2. The results shown below represent averages from all runs.

We discuss two types of wireless links. *HSCSD* links simulate the *High Speed Circuit Switched Data* service of GSM, the European Cellular standard. HSCSD bundles multiple circuit switched GSM links together to increase bandwidth. The HSCSD links simulated have a bandwidth of 86.4 Kbps and use 100 byte frames. To reduce losses during *fade* periods bit interleaving is used [9], simulated by a 100 ms delay. Bit interleaving randomizes losses [9], so we used independent frame losses at rates of 1%, 2%, 5% and 10%. *WLAN* links simulate an IEEE 802.11b WLAN with 5 Mbps of bandwidth and a 3 ms delay, using 1000 byte frames. To allow comparisons with previous studies, WLAN links corrupt bits at exponentially distributed intervals with average durations of 2^{14}, 2^{15}, 2^{16} or 2^{17} bits [12]. Table I summarizes these parameters, showing the actual frame loss rates measured during our tests, which are very close to analytical predictions. The error processes in each link direction were identical but independent. We ran tests under error free conditions for reference purposes. We ignored TCP, UDP and IP headers as they uniformly influence all link layer schemes. In contrast, we accounted for the *exact* amount of framing overhead required by each scheme.

For each wireless link we simulated the topologies depicted in Fig. 1. In the two wireless link topology (solid frame) Wireless Host A communicates with Wireless Host B via a wired link. In each test, both wireless links were of the same type but independent of each other. This topology simulates peer-to-peer communication, with both peers on wireless access networks. In the one wireless link topology (dotted frame), Base Station A communicates with Wireless Host B, again via a wired link. Base Station A is the "server," i.e. most data flows in the wired to wireless direction. Data may still flow in the reverse direction, for example TCP ACKs and interactive user input. This topology simulates client-server communication, with the client on a wireless access network. For WLAN tests the wired link was a 10 Mbps LAN with 1 ms delay, simulating a departmental network, while for HSCSD tests it was a 2 Mbps WAN with 50 ms delay, simulating a long Internet path.

We tested both TCP and UDP applications to evaluate the suitability of various link layer schemes for each application class. For TCP, we simulated large file transfers and World Wide Web browsing, using the TCP Reno module of ns-2 with 500 ms granularity timers. For UDP, we simulated continuous media distribution, a delay sensitive but error tolerant application. To establish a baseline we simulated a *Raw Link* scheme, which does not perform any link layer error recovery, under both TCP and UDP.

For TCP we tested reliable schemes delivering frames in sequence to higher layers to avoid TCP retransmissions. *Go Back N* is a basic sliding window scheme, i.e. the sender buffers outgoing frames and retransmits unacknowledged frames after a timeout. The receiver positively acknowledges frames received in sequence, dropping out of sequence ones. After a timeout the sender retransmits *all* outstanding frames. *Selective Repeat* improves upon this by buffering out of sequence frames

Link type	Bandwidth	Delay	Frame size	Loss model	Measured frame loss rates			
HSCSD	86.4 Kbps	100 ms	100 Bytes	Independent, frame based	1%	2%	5%	10%
WLAN	5 Mbps	3 ms	1000 Bytes	Exponential, bit based	0.8%	1.5%	3%	5.9%

TABLE I
SIMULATED CHARACTERISTICS FOR EACH WIRELESS LINK

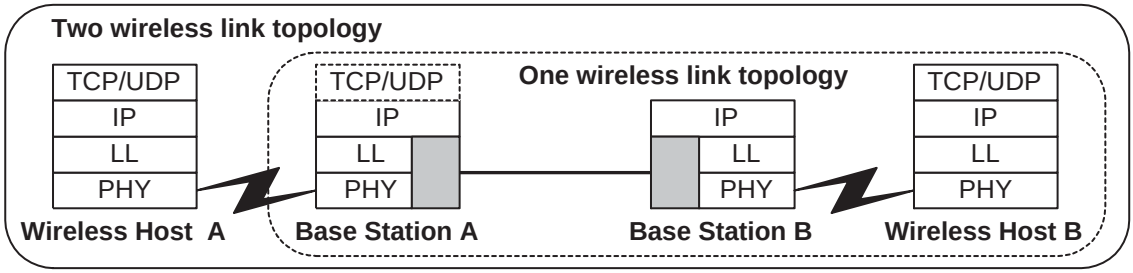

Fig. 1. One and two wireless link simulation topologies

at the receiver and returning *negative* ACKs (NACKs) when it detects gaps in the sequence, thus allowing the sender to retransmit only lost frames. We used a Selective Repeat variant allowing multiple NACKs per loss [15]. In both schemes, under persistent losses the sender may exhaust its window and stall, retransmitting the same frame indefinitely. Each frame includes sequence and acknowledgment numbers (2 bytes).

To prevent conflicts with TCP retransmissions [11], the sender in *Karn's RLP* abandons frames not received after some (by default 3) retransmissions [9]. Thus, delay over the link is bounded and the sender does not stall. This scheme uses only NACK and *keepalive* messages during idle periods, thus frames only include a sequence number (1 byte). *Berkeley Snoop* is a TCP aware scheme [12], provided by ns-2. A module at the wireless base station *snoops* inside TCP segments, buffering data sent to the wireless host. If duplicate TCP ACKs indicate a lost packet, this is retransmitted by the base station and the ACKs are suppressed.

For the UDP-based continuous media distribution application, low delay is preferred over full reliability. *Forward Error Correction* (FEC) schemes offer limited recovery by adding redundancy to the transmitted stream, allowing the receiver to recover from losses. The *XOR based FEC* scheme sends each data frame unmodified, but every 8 (for HSCSD) or 12 (for WLAN) frames a *parity* frame is also transmitted, generated by XOR'ing the preceding (8 or 12) data frames, collectively called a *block*. If a *single* data frame is lost from a block, we can recover it by XOR'ing the remaining data with the parity frame. A timeout is used to emit a parity frame when the link becomes idle during a block. All frames include a sequence number (1 byte).

The UDP-based application was tested with Selective Repeat to examine the interactions between a fully reliable scheme and a delay sensitive application. We also tested Karn's RLP, using 1 retransmission per loss to keep delay low. In this scheme, frame losses cause subsequently received frames to wait until the missing one is received or abandoned. This is critical for TCP but detrimental to applications with their own resequencing (playback) buffers. Our *Out of Sequence* (OOS) variant of Karn's RLP immediately releases all received frames to avoid such delays.

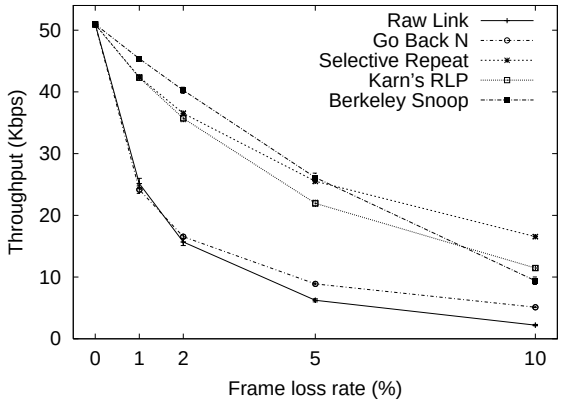

Fig. 2. FTP throughput, one HSCSD link

IV. FILE TRANSFER

The first TCP-based application tested was file transfer over FTP. We sent a file from a wireless or wired server to a wireless client (see Section III). File transfers are unidirectional, with TCP ACKs moving in the reverse direction. While longer transfers produce more stable results, users do not make infinite transfers, so we used file sizes of 10 MBytes for HSCSD and 100 MBytes for WLAN tests. The ns-2 FTP module sends data as fast as possible; TCP handles flow and congestion control. As TCP completely controls FTP behavior, performance studies usually rely on FTP transfers in the wireless server to wired client direction in order to characterize TCP performance [12]. We measured application throughput, i.e. the amount of *application* data transferred divided by total time. TCP and link layer retransmissions are *not* included as they signify overhead.

Fig. 2 shows FTP performance in the one HSCSD link scenario for various frame loss rates. Each curve depicts appli-

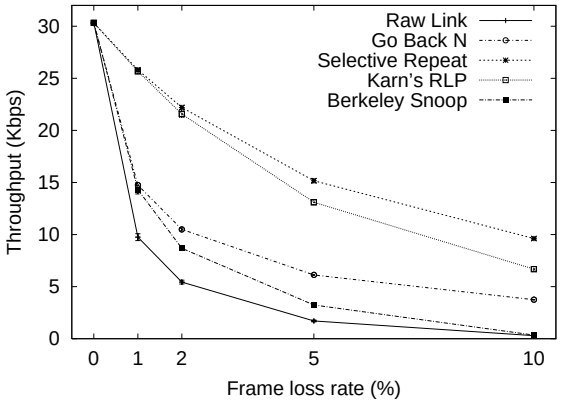

Fig. 3. FTP throughput, two HSCSD links

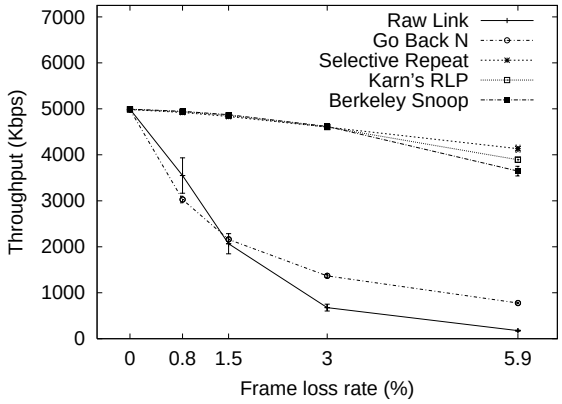

Fig. 4. FTP throughput, one WLAN link

cation throughput for a particular link layer scheme, averaged among 30 test repetitions, with error bars at plus/minus one standard deviation. Raw Link clearly illustrates how wireless losses affect TCP performance: by increasing the frame loss rate from 1% to 2%, throughput drops by 30%. Berkeley Snoop performs best in most cases, with Karn's RLP and Selective Repeat lagging behind, due to their 1 or 2 byte link layer overhead, a significant factor for 100 byte frames. At higher loss rates Selective Repeat is better than Karn's RLP due to its persistent loss recovery, despite its higher overhead. Berkeley Snoop deteriorates at high loss rates due to the loss of *duplicate* TCP ACKs. Since TCP ACKs are cumulative, rare losses are not critical, but frequent losses prevent loss detection and local recovery. Go Back N is worse than Raw Link at low loss rates, with minor improvements as conditions deteriorate. These results suggest that conflicting TCP and link layer retransmissions [11] are less of a problem than resorting to TCP retransmissions, unless if naive schemes such as Go Back N are used.

The limitations of Berkeley Snoop become clear in the two HSCSD link scenario, where both client and server are wireless. Fig. 3 shows that with two wireless links FTP performance over Raw Link drops to less than 40% of that in the previous scenario, indicating that the effects of multiple wireless links are multiplicative. Since Berkeley Snoop only retransmits from the base station to the wireless host, it cannot retransmit TCP data over both wireless links in the path, thus resorting to TCP recovery and performing worse than Go Back N. This occurs in *any* file transfer *from* a wireless host. In contrast, Selective Repeat and Karn's RLP offer significant gains since they retransmit in both directions. In the two WLAN link scenario, the results are similar.

Fig. 4 shows FTP throughput in the one WLAN link scenario. Selective Repeat, Karn's RLP and Berkeley Snoop perform nearly the same since their link layer overhead is insignificant with 1000 byte frames. At the highest loss rates tested, the most persistent error recovery schemes perform better, therefore Selective Repeat is ahead of Karn's RLP, which is ahead of Berkeley Snoop. Go Back N performs either worse or marginally better than Raw Link. Note that even though the end-to-end delay is mostly determined by the wireless links, the coarse grained TCP timers make TCP recovery much slower than link layer recovery.

Overall, our FTP tests show that the more advanced TCP unaware link layer schemes provide significant throughput improvements over plain TCP. They also suggest that conflicting TCP and link layer retransmissions are less of a problem than resorting to end-to-end TCP recovery. Berkeley Snoop faces problems with multiple wireless links and transfers from wireless hosts. Actually, these problems are inherent in *any* TCP aware scheme employing *only* TCP ACKs for loss detection, as this requires the data receiver to be close to the base station. In contrast, TCP unaware recovery schemes use their own ACKs, hence they are independent of network topology and file transfer direction.

V. World Wide Web Browsing

Most studies of wireless TCP performance employ large file transfers since they are easy to simulate, their performance metric (throughput) eventually converges and they directly reflect TCP behavior. A single wireless link topology is usually considered, with data flowing from a wired server to a wireless client. Since a TCP application can be viewed as a set of file transfers, it is assumed that FTP throughput in this scenario adequately characterizes TCP performance [12]. This generalization however is flawed. Real applications make many *short* data exchanges, thus TCP rarely reaches its peak FTP throughput. In addition, most applications are either interactive, e.g. Telnet, or employ request/reply protocols, e.g. SMTP, thus data flows in *both* directions, and *each* exchange must complete for the application to proceed, regardless of its direction and size.

Therefore, for a more realistic view of TCP application performance we simulated *World Wide Web* (WWW) browsing over HTTP [16], the most popular Internet application. A WWW client accesses *pages* containing text, links and embedded objects, stored on a WWW server. The client-server interaction consists of *transactions*: the client requests a page from a server, the server returns the page, complete with pointers to its embedded objects, the client requests each object, and the server returns them, completing the transaction. All transfers are performed over TCP.

The ns-2 HTTP module provides empirical distributions for the request, page and embedded object sizes, as well as the

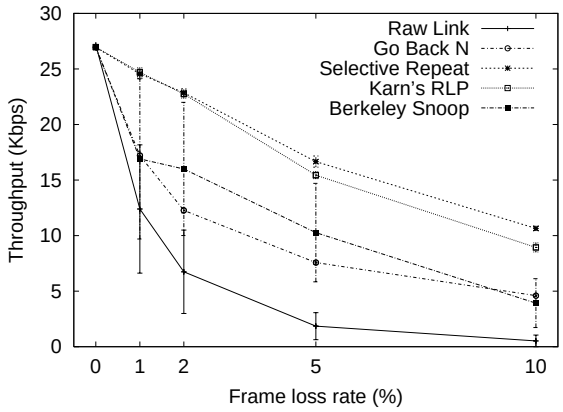

Fig. 5. HTTP throughput, one HSCSD link

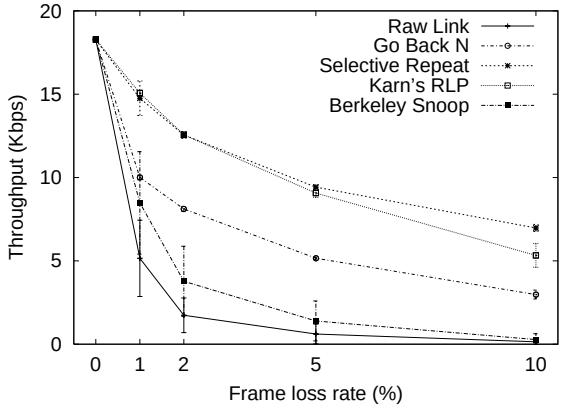

Fig. 6. HTTP throughput, two HSCSD links

number of objects per page [16]. Only one transaction is in progress at any time and there are no pauses between transactions. WWW browsing was simulated between a wired or wireless server and a wireless client (see Section III) for 2000 s (HSCSD) or 500 s (WLAN). We measured WWW browsing throughput, i.e. the amount of *application* data transferred from the server to the client, including both pages and embedded objects, divided by total time. Only completed transactions were included. Client requests influence throughput indirectly, by introducing delays.

Fig. 5 shows WWW browsing throughput in the one HSCSD link scenario. Raw Link performance drops dramatically as losses increase. Karn's RLP and Selective Repeat provide significant gains throughout the frame loss range. While Selective Repeat is slightly worse than Karn's RLP at low loss rates due to its increased overhead, at higher loss rates its persistence makes it a better choice. On the other hand, Berkeley Snoop performs only slightly better than Go Back N, with both performing considerably worse than Selective Repeat and Karn's RLP. Berkeley Snoop offers moderate gains over Raw Link, even though most data flows in the wired to wireless direction, due to its inability to recover from losses in the reverse direction.

These results illustrate that unidirectional error recovery is *insufficient* for interactive applications, since client requests, although short, are critical for performance. HTTP and FTP

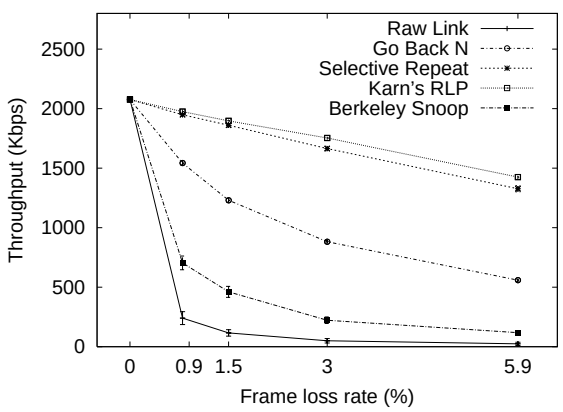

Fig. 7. HTTP throughput, one WLAN link

throughput results are *not* comparable, since WWW browsing throughput incorporates client request delays. WWW browsing throughput in the two HSCSD link scenario, shown in Fig. 6, reveals a similar picture. The only difference is in Berkeley Snoop, which is closer to Raw Link, as in this case it has problems in both wireless links.

Fig. 7 shows WWW browsing throughput in the one WLAN link scenario. Interestingly, in this scenario Karn's RLP beats Selective Repeat at higher loss rates. This is due to the reliability of the link, which makes Selective Repeat's persistence less important, and to the short HTTP transfers: when the link is idle, Selective Repeat does not receive ACKs when the last packet sent is lost, thus resorting to timeouts, while Karn's RLP sends *keepalives* when idle, thus triggering NACKs. Berkeley Snoop is close to Raw Link, as it is unable to recover from lost client requests, while Go Back N is roughly in the middle. WWW browsing results from the two WLAN link scenario are similar.

Overall, our HTTP tests show that the advanced TCP unaware schemes offer considerable improvements in both one and two wireless link scenarios, while Berkeley Snoop is unable to similarly improve WWW browsing performance, even under its most favorable scenario for FTP tests. Our results again suggest that conflicting TCP and link layer retransmissions are less of a problem than resorting to end-to-end TCP recovery. The most important conclusion however is that the short bidirectional transfers of interactive applications considerably differentiate them from unidirectional FTP transfers. Therefore, schemes based on unidirectional recovery, such as Berkeley Snoop, are inherently unable to improve the performance of these applications.

VI. Continuous Media Distribution

Applications that use UDP due to its simplicity, such as NFS, would find acceptable the reliable link layer schemes discussed above, but delay sensitive applications that use UDP in order to handle flow and congestion control themselves are a poor match for Selective Repeat which may stall with persistent losses. We tested UDP performance using real-time continuous media distribution, i.e. a lecture where a speaker sends audio, and possibly video, to an audience including a wireless client. We

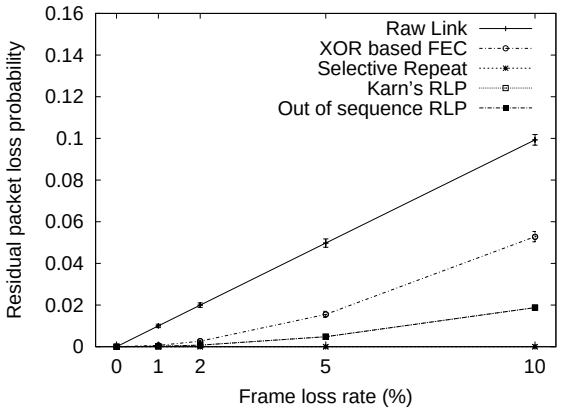

Fig. 8. CBR loss, one HSCSD link

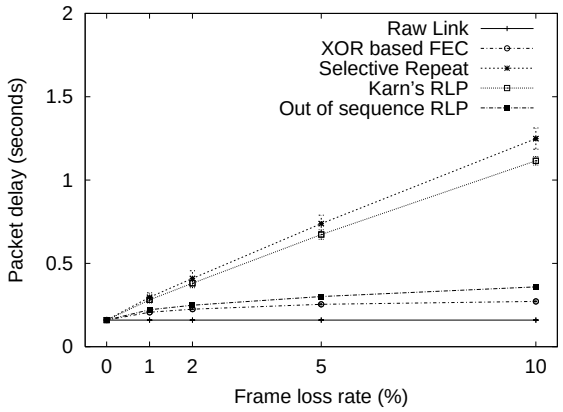

Fig. 9. CBR delay, one HSCSD link

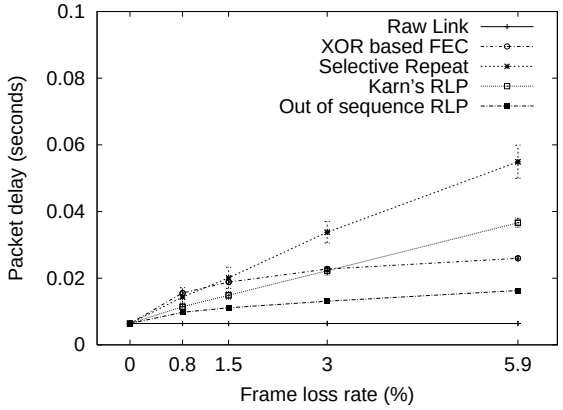

Fig. 10. CBR delay, one WLAN link

used a speech model where the speaker alternates between *talking* and *silent* states with exponential durations, averaging 1 s and 1.35 s, respectively [17]. Media are only transmitted in the talking state. When talking, the speaker transmits packets isochronously at a *Constant Bit Rate* (CBR) of 14.4 Kbps for HSCSD (speech) or 1 Mbps for WLAN (audio/video).

We assumed that the CBR application uses FEC to tolerate some loss without end-to-end retransmissions, thus enabling the sender to handle multiple recipients. Received packets are buffered until a *playback point* determined by human perception, thus supporting smooth playback despite delay variations. Such measures are necessary to handle the variable delays and losses due to Internet congestion, but they are insufficient for wireless losses. To characterize CBR performance, we measured the *residual* loss rate at the receiver after link layer recovery. Since packets missing the playback point are dropped, loss rate is coupled with a delay metric covering *most* packets. We used mean packet delay plus twice its standard deviation to incorporate the variable delays introduced by each scheme. We ran each CBR test for 2000 s (HSCSD) or 500 s (WLAN).

Fig. 8 depicts residual loss for the one HSCSD link scenario. The transfer direction is unimportant, since all simulated schemes are symmetric. Raw Link exhibits the native loss rate, closely matching nominal rates. XOR based FEC, with 1 parity for 8 data frames over HSCSD (or 12 data frames over WLAN), does not reduce losses in proportion to its overhead, e.g. a 10% native loss rate is reduced to 5.4% by adding 12.5% of overhead, since the parity frame is wasted both when no losses occur and when more than one loss occurs. Selective Repeat achieves full recovery, since it always retransmits lost packets, albeit at the cost of very high delays. Karn's RLP and our *Out of Sequence* (OOS) RLP, with 1 retransmission per loss, perform identically, since their recovery mechanism is the same. Both RLP variants considerably outperform the FEC scheme, while also introducing less overhead. Even though limited recovery does not eliminate losses, if it keeps losses low enough for an error tolerant application, there is no need to resort to full recovery. The loss results for the other scenarios are similar.

Delay results for the one HSCSD link scenario are shown in Fig. 9. Note that we first calculated the delay metric for *each* test repetition and then calculated the overall average. Raw Link exhibits the fixed native delay of the path. Selective Repeat suffers from the highest delay, since correctly received frames may be delayed after losses so as to achieve in sequence delivery and the sender may also stall due to persistent losses. Karn's RLP is slightly faster than Selective Repeat as it abandons recovery after a single failed retransmission. The lowest delay among retransmission based schemes is provided by OOS RLP which releases all received frames immediately. XOR based FEC, which also releases recovered frames out of sequence, is slightly faster than OOS RLP as it avoids retransmissions over the high delay HSCSD link. The FEC scheme uses a timeout of twice the regular packet interval to prematurely terminate blocks when the sender becomes silent. Loss and delay results for the two HSCSD link scenario are very similar.

Delay results for the one WLAN link scenario, given in Fig. 10, shows OOS RLP and, for low loss rates, even Karn's RLP to be faster than XOR based FEC. To understand why, consider how error recovery works. When a loss is detected, both RLP schemes send a NACK which triggers a retransmission, thus recovery takes a round trip delay plus a frame transmission delay. In the FEC scheme, the current block must complete before recovery, thus, on average, recovery takes the transmission delay of half a block. WLAN links have low delays and use large frames, hence retransmissions are *faster* than block based

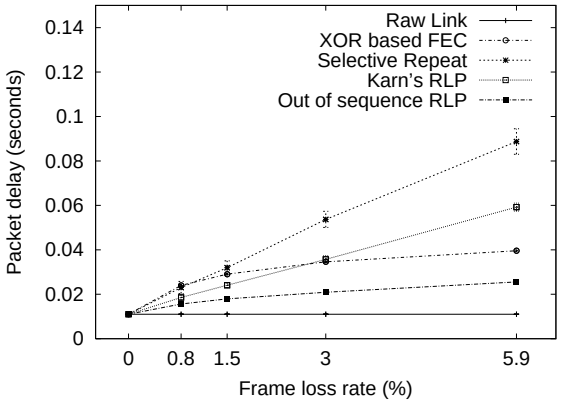

Fig. 11. CBR delay, two WLAN links

FEC. Again, Selective Repeat exhibits the highest delay metrics of all schemes.

Fig. 11 shows delay for the two WLAN link scenario. While all curves have the same shape as above, with two WLAN links the delay metric for Karn's RLP increases between 61.7% and 62.4%, while for OOS RLP it increases between 56.7% and 59.8%. Therefore, with two WLAN links the delay for Karn's RLP increases at a *faster* rate. This is because with two wireless links Karn's RLP may delay a frame *repeatedly* due to unrelated losses, while OOS RLP delays only the retransmitted frames. Thus, out of sequence delivery becomes increasingly valuable on paths with multiple wireless links, even for playback applications.

Overall, our UDP tests show that the RLP schemes adequately reduce residual losses for loss tolerant applications, unlike FEC whose gains do not justify its overhead and Selective Repeat whose full recovery makes it unsuitable for delay sensitive applications. Our OOS RLP variant reduces delay over Karn's RLP, with increased gains over multiple wireless links. Thus, it is a good match for playback applications such as continuous media distribution. OOS RLP also turns out to be faster than block based FEC in low delay links with long frames and large blocks. Thus, to select a scheme we must take into account the underlying link.

VII. Conclusions

We have presented a comprehensive simulations of diverse TCP and UDP-based applications, using a variety of wireless links and link layer schemes. Based on these results we can draw a number of conclusions. First, file transfers are an inadequate model for interactive applications, as evidenced by WWW browsing. Since the majority of TCP traffic is interactive, this is our most important conclusion. Second, both file transfer and WWW browsing tests suggest that retransmission conflicts between TCP and the link layer are not a serious problem. Third, TCP unaware link layer schemes performed excellent for both file transfer and WWW browsing, regardless of topology, unlike TCP aware schemes. Fourth, the same schemes performed best for both file transfer and WWW browsing, therefore the performance of diverse TCP-based applications can be improved by a single scheme. Fifth, the schemes that performed best for TCP-based applications were different from those that performed best for the UDP-based application.

Furthermore, our results extend conclusions stated in previous work [18], by extending past simulations with additional wireless links (HSCSD and WLAN), topologies (both WAN and LAN) and link layer schemes (Go Back N for TCP and Selective Repeat for UDP). First, for the stringent delay requirements of real-time applications fast recovery can also be provided by limited retransmission schemes, in addition to conventional FEC schemes. Second, for high speed and low delay links limited retransmissions may be *faster* than FEC. Third, our out of sequence limited recovery scheme (OOS RLP) combines efficiency with low delay and is a good match for playback applications.

References

[1] G. Xylomenos and G. C. Polyzos, "Internet protocol performance over networks with wireless links," *IEEE Network*, vol. 13, no. 5, pp. 55–63, July/August 1999.

[2] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," RFC 2001, January 1997.

[3] R. Ludwig and R. H. Katz, "The Eifel algorithm: making TCP robust against spurious retransmissions," *Computer Communications Review*, vol. 30, no. 1, pp. 30–36, January 2000.

[4] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," RFC 2018, October 1996.

[5] B. R. Badrinath, A. Bakre, T. Imielinski, and R. Marantz, "Handling mobile clients: A case for indirect interaction," in *Proc. of the 4th Workshop on Workstation Operating Systems*, 1993, pp. 91–97.

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

[7] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proc. of the IEEE INFOCOM '00*, 2000, pp. 1537–1545.

[8] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, July 2001.

[9] P. Karn, "The Qualcomm CDMA digital cellular system," in *Proc. of the USENIX Mobile and Location-Independent Computing Symposium*, 1993, pp. 35–39.

[10] S. Nanda, R. Eljak, and B. T. Doshi, "A retransmission scheme for circuit-mode data on wireless links," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 8, pp. 1338–1352, October 1994.

[11] A. DeSimone, M. C. Chuah, and O. C. Yue, "Throughput performance of transport-layer protocols over wireless LANs," in *Proc. of the IEEE GLOBECOM '93*, 1993, pp. 542–549.

[12] 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*, 1996, pp. 256–267.

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

[14] "Multi service link layer extensions for ns-2," Available at http://www.mm.aueb.gr/~xgeorge/codes/codephen.htm.

[15] P. T. Brady, "Evaluation of multireject, selective reject, and other protocol enhancements," *IEEE Transactions on Communications*, vol. 35, no. 6, pp. 659–666, June 1987.

[16] B. A. Mah, "An empirical model of HTTP network traffic," in *Proc. of the IEEE INFOCOM '97*, 1997, pp. 592–600.

[17] S. Nanda, D. J. Goodman, and U. Timor, "Performance of PRMA: a packet voice protocol for cellular systems," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 3, pp. 584–598, August 1991.

[18] G. Xylomenos and G.C. Polyzos, "Quality of service support over multi-service wireless internet links," *Computer Networks*, vol. 37, no. 5, pp. 601–615, 2001.