A multi-service link layer architecture for the wireless Internet

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

***Abstract*— The performance of Internet applications over wireless links is disappointing due to the adverse effects of wireless errors on higher layer protocols and applications. This paper focuses on link layer enhancement mechanisms which attempt to hide these wireless impairments. We simulate 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 behaves differently than bulk file transfer, that some TCP aware enhancements have limited applicability and that UDP applications are best served by schemes inappropriate for TCP. We then describe a multi-service link layer architecture that simultaneously enhances the performance of diverse applications by supporting multiple error recovery schemes in parallel. In order to evaluate our architecture, we repeat our previous simulations with all applications executing simultaneously. The results reveal that with our approach each application achieves similar improvements as when it operates alone over its preferred link layer.**

I. INTRODUCTION

The explosive growth of the Internet is only paralleled by the growth in wireless communications. *Cellular Telephony* is evolving towards higher bit rates, while *Wireless Local Area Networks* (WLANs) have become commodity items. Even though these links lag behind wired ones in performance, their popularity makes their integration into the Internet very important. Unfortunately, higher layers make assumptions about link performance that cannot always be met by wireless links. Thus, although supporting IP over these links is straightforward, their performance can be disappointing [22]. As the Internet evolves towards *Quality of Service* (QoS) provision so as to support applications requiring service guarantees, improving the performance of Internet applications over wireless links becomes critical.

This paper evaluates link layer mechanisms that aim to improve the performance of Internet applications over wireless links and describes an architecture that simultaneously provides the benefits of multiple such mechanisms. In Section II we outline the problem and review related work. Section III describes the single application simulation setup and the performance of three applications: file transfer, World Wide Web browsing and continuous media distribution. Our results reveal that UDP applications favor different enhancements than TCP ones, so in Section IV we describe a multi-service link layer approach which supports multiple link layer mechanisms. Section V describes the multiple application simulation setup and the performance of all three applications executing in parallel. Our results reveal that application performance improves by similar factors as when each application operates alone over its preferred scheme.

Corresponding Authos: George Xylomenos, Dept. of Informatics, Athens University of Economics and Business, Patision 76, Athens 104 34, Greece. Tel: +30-210-8203591. Fax: +30-1-8203686.

II. BACKGROUND AND RELATED WORK

IP offers an unreliable packet delivery service, meaning that packets may be lost, reordered or duplicated. Applications can use UDP for (nearly) direct access to this service when they expect the network to be reliable enough for their needs. For example, file sharing over wired LANs usually employs UDP, coupled with a simple application level retransmission policy, due to the high reliability of these networks. Another motivation to use UDP is sensitivity to delay. For example, real-time conferencing applications usually employ UDP, even over WANs, adding redundancy to their data to tolerate congestion losses without retransmissions.

Most applications however prefer delegating error recovery to the transport layer, hence they employ TCP which offers a reliable byte stream service. TCP segments the application data stream into IP packets and reassembles it at the receiver. The receiver generates cumulative *acknowledgments* (ACKs) for segments received in sequence, returning duplicates of the last ACK 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. Due to the high reliability of wired links, TCP assumes that all losses are due to congestion, thus after a loss it reduces its transmission rate to relieve congestion and then gradually increases it so as to probe the network [21].

We focus on the widely available Cellular and WLAN systems. Their reduced reliability causes UDP application performance to degrade or even become unacceptable, since UDP does not provide error recovery and application level error recovery is only sufficient for congestion losses. On the other hand, wireless errors cause TCP to reduce its transmission rate to avoid what it (falsely) regards as congestion, leading to dramatic throughput reductions.

The performance of UDP applications over wireless links has not been studied extensively, as their diverse behaviors

and error recovery strategies prevent formulation of a unified model. Furthermore, UDP applications are usually perceived as LAN oriented, a situation challenged by multimedia streaming on the Internet. Considerable work has been devoted however to TCP. Generic TCP enhancements such as *Eifel* [13] and *Selective Acknowledgments* [16] improve TCP performance by avoiding redundant TCP retransmissions after a loss. These approaches however do not avoid the inherent delay of end-to-end retransmissions.

Wireless TCP enhancement schemes try to avoid triggering congestion recovery when wireless errors occur. One approach is to *split* TCP connections into one connection over the wireless link and another one over the wired part of the path [3]. Error recovery is only performed over the wireless link. This violates the end-to-end semantics of TCP and is incompatible with IP security which encrypts TCP headers [11]. Another approach is to *freeze* TCP state whenever persistent errors occur [8], [12]. While error conditions persist, TCP does not invoke its congestion control mechanisms. Wireless and congestion losses may be differentiated either via explicit loss notifications [8] or explicit congestion notifications [12]. Thus performance is not unnecessarily degraded, but recovery remains end-to-end and protocol software must be modified at various locations so as to generate these notifications.

The alternative to TCP modifications is link layer error recovery over the wireless link, as in the *Radio Link Protocols* (RLPs) of Cellular systems [10], [17]. One drawback of RLPs is that they were designed for reliable data exchange, hence they apply schemes which may be inappropriate for some traffic. For example, retransmissions delay real-time traffic, so they should be bypassed for non-TCP traffic [14], [20]. Another problem is that RLP recovery may interfere with TCP recovery [7], leading to conflicting retransmissions between the link and transport layers, at least if the link layer delay is comparable to the end-to-end delay.

Another option is to exploit transport layer information at the link layer. By *snooping* inside *each* TCP stream at the wireless base station we can retransmit lost segments when duplicate ACKs arrive, hiding these duplicates from the sender to avoid end-to-end recovery. This approach reduces overhead as it avoids link layer control messages and it prevents cross layer interactions [4], but it is incompatible with IP security as it examines TCP headers.

Link layer approaches have the advantage of requiring only local deployment over the wireless link [19], without changes to higher layers. This makes them transparent to the rest of the Internet, enables them to recover faster than transport layer schemes and allows them to customize recovery to the underlying link. Higher layer approaches have the advantage of providing solutions customized to application requirements. Our work combines the advantages of both approaches, by providing local recovery for diverse applications.

III. SINGLE APPLICATION PERFORMANCE

A. Simulation setup

To study the performance of Internet applications over various link layer schemes and wireless link types, we performed extensive simulations using ns-2 [2], extended with additional wireless links, link layers and application models [1]. To compensate for statistical fluctuations, each test was repeated 30 times. All results shown represent average values.

We present results for two types of wireless link. *HSCSD* links simulate the *High Speed Circuit Switched Data* service of GSM, the European cellular standard. HSCSD bundles multiple circuit switched GSM links 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, simulated by an increased 100 ms delay. Bit interleaving randomizes losses [10], so we used an independent frame loss model at rates of 1%, 2%, 5% and 10%. The *WLAN* links simulate the behavior of 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 [4]. Table I summarizes these parameters, also showing the measured frame loss rates, which are very close to analytical calculations. The error processes in each link direction were identical but independent. We ignored TCP, UDP and IP headers as they uniformly influence all link layer schemes, but accounted for the *exact* framing overhead of each scheme.

For each type of wireless link we simulated the two topologies of Fig. 1. In the two wireless link topology (solid connection) Wireless Host A communicates with Wireless Host B via a wired link. In this (symmetric) configuration both wireless links are of the same type (HSCSD or WLAN) 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 connection), Wired Host A communicates with Wireless Host B, again via a wired link. In this (asymmetric) configuration Wired Host A is the server and Wireless Host B is the client. Most data flows in the wired to wireless direction, with some traffic in the reverse direction, such as TCP ACKs or user input in interactive applications. This topology simulates client-server communication, with the client on a wireless access network. For WLAN tests the wired link is a 10 Mbps LAN with 1 ms delay, simulating a departmental network, while for HSCSD tests it is a 2 Mbps WAN pipe with 50 ms delay, simulating a long Internet path.

To evaluate the suitability of each link layer scheme for each application, we tested both TCP and UDP applications operating in isolation [24]. For TCP, we simulated file transfer and World Wide Web browsing, using TCP Reno with 500 ms granularity timers. For UDP, we simulated continuous media distribution, a delay sensitive but moderately error tolerant application. All applications are described below. To establish a baseline, we simulated a *Raw Link* scheme which performs no error recovery, with both TCP and UDP applications.

For TCP we focused on reliable schemes that deliver frames in sequence to higher layers, so as to avoid triggering 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

TABLE I

SIMULATED CHARACTERISTICS FOR EACH WIRELESS LINK TYPE

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

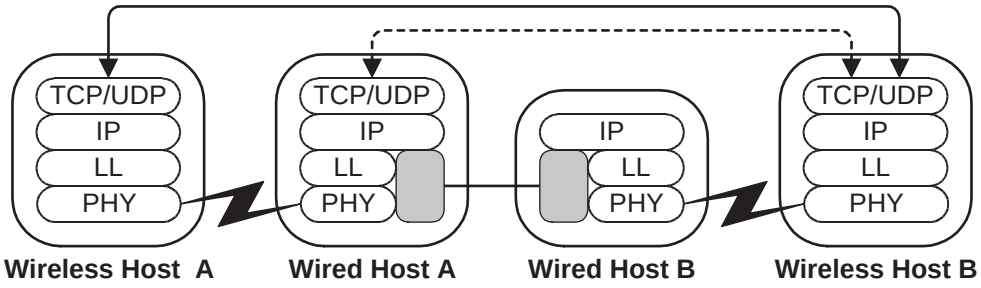

Fig. 1. Simulation topologies

acknowledges frames received in sequence and drops everything else. Thus, after a timeout the sender must retransmit *all* outstanding frames. *Selective Repeat* improves this approach by buffering out of sequence frames at the receiver and returning *negative* ACKs (NACKs) when it detects gaps in the sequence space, thus allowing the sender to retransmit only lost frames. We used a Selective Repeat variant allowing multiple NACKs per loss [6]. In both schemes each frame includes sequence and acknowledgment numbers (2 bytes).

With these schemes the sender may stall under persistent losses, retransmitting the same frame forever. To prevent conflicts with TCP retransmissions in this case [7], in *Karn's RLP* the sender abandons frames not received after some (by default 3) retransmissions [10]. Thus, delay over the link is bounded and the sender never stalls, given a sufficient window. In this scheme only NACK and *keepalive* messages during idle periods are needed, thus frames include only sequence numbers (1 byte). *Berkeley Snoop* is a TCP aware scheme [4]. A module at the wireless base station *snoops* inside TCP segments, buffering data sent *to* the wireless host. If duplicate TCP ACKs indicate that a packet was lost, it is retransmitted by the base station and the ACKs are suppressed. This scheme does not retransmit in the direction *from* the wireless host. There is no frame overhead with Berkeley Snoop.

For UDP we focused on schemes that favor low delay over full reliability. *Forward Error Correction* (FEC) schemes add error recovery overhead to the transmitted stream, allowing the receiver to recover from some losses. The *XOR based FEC* scheme sends data frames unmodified, but every 8 (HSCSD) or 12 (WLAN) frames an extra *parity* frame is transmitted, constructed by XOR'ing the preceding (8 or 12) data frames, collectively called a *block*. If exactly *one* 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 application was also tested with Selective Repeat to examine its behavior over a fully reliable scheme.

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, so as to deliver received frames in sequence. While this is critical for TCP, it is detrimental to applications that use their own resequencing (playback) buffers, especially over paths with multiple wireless links. Therefore, we also tested our *Out of Sequence* (OOS) RLP [25] which modifies Karn's RLP by immediately delivering received frames.

B. File transfer

The first TCP application tested was file transfer (FTP). We simulated a file transfer from a wireless or wired server to a wireless client. File transfers are unidirectional, with only TCP ACKs in the reverse direction. FTP sends data as fast as possible, with TCP handling flow and congestion control. While longer transfers produce more stable results, in practice users do not initiate huge transfers. Thus, we used 2 MByte files for HSCSD and 100 MByte files for WLAN tests. We measured application throughput, defined as the amount of *application* data transferred divided by time taken. Note that retransmissions are *not* included.

Fig. 2 shows FTP throughput with one WLAN link for a range of frame loss rates. Each curve depicts application throughput for the link layer scheme indicated, averaged among 30 test repetitions, with error bars at plus/minus one standard deviation. The Raw Link scheme illustrates the dramatic impact of wireless losses on TCP: by increasing frame losses from 0.8% to 1.5%, throughput drops by 30%. Selective Repeat, Karn's RLP and Berkeley Snoop perform nearly the same since they can all handle such modest loss rates, and their different link layer overhead is insignificant for the large frames used. 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

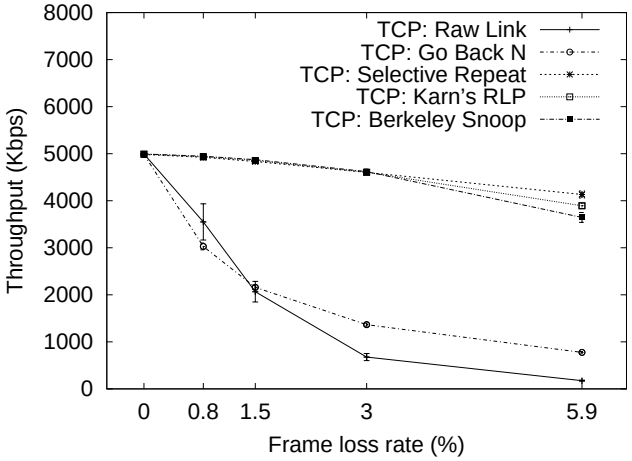

Fig. 2. File transfer throughput, one WLAN link

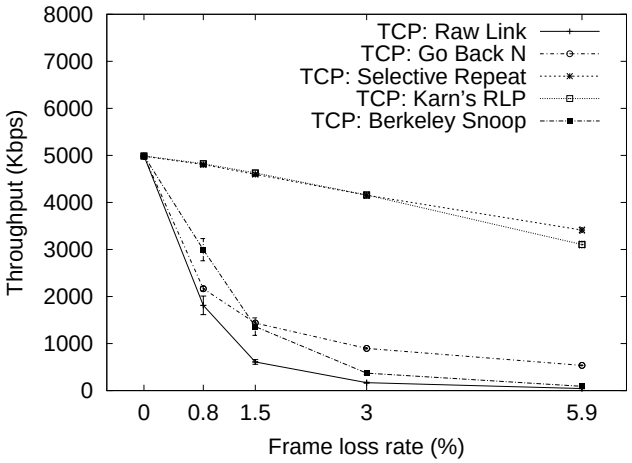

Fig. 3. File transfer throughput, two WLAN links

ahead of Berkeley Snoop. Go Back N however is at best a marginal improvement over Raw Link. These results suggest that conflicting TCP and link layer retransmissions [7] are less of a problem than resorting to end-to-end TCP retransmissions.

The limitations of Berkeley Snoop become clear in the two WLAN link scenario, where both the client and the server are wireless, as shown in Fig. 3. Berkeley Snoop only works in the direction from the wired Internet towards a wireless host due to its reliance on TCP ACKs. In the reverse direction, TCP ACKs are returned late and may even signify congestion losses [22]. Since only the base station makes retransmissions towards the wireless host, TCP data cannot be retransmitted over both wireless links in the path, thus we resort to TCP recovery for one of them. Therefore, Berkeley Snoop provides only minor improvements over Raw Link, similar to those of Go Back N. In contrast, Selective Repeat and Karn's RLP offer significant gains since they retransmit in both directions. This situation also arises with file transfers from a wireless to a wired host. Note that even though in this scenario the end-to-end delay is low, the coarse grained TCP timers make TCP recovery much slower than link layer recovery.

Results with one HSCSD link, shown in Fig. 4, reveal that

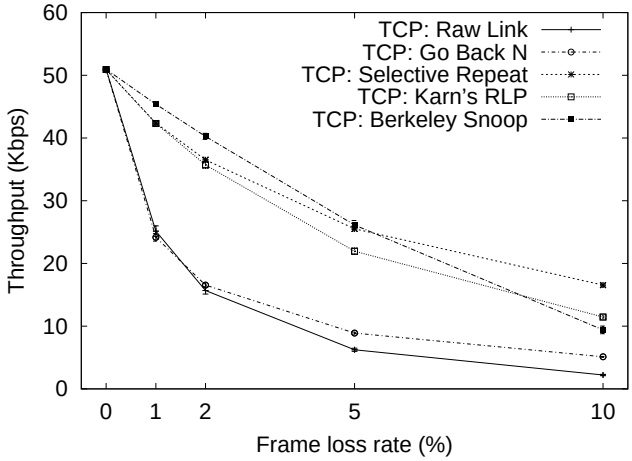

Fig. 4. File transfer throughput, one HSCSD link

Berkeley Snoop performs best at most frame loss rates, with Karn's RLP and Selective Repeat lagging behind it. This is due to their 1 or 2 byte link layer overhead, a significant factor for the small frames used. Berkeley Snoop deteriorates at higher 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 retransmissions. The two HSCSD link case is similar to the two WLAN link case.

Overall, our FTP tests reveal that some TCP unaware link layer schemes (Selective Repeat and Karn's RLP) provide significant throughput improvements over plain TCP (Raw Link). 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 severe problems with multiple wireless links and transfers from wireless hosts. These are inherent in *any* TCP aware scheme employing *only* TCP ACKs for loss detection, as this implicitly requires the client to be next to the base station. TCP unaware recovery schemes use their own ACKs, which do incur some overhead but make them independent of both network topology and file transfer direction.

C. World Wide Web browsing

As TCP completely controls FTP behavior, most studies of wireless TCP performance focus on large file transfers which are easy to simulate and summarize with a simple metric (throughput). Usually, only a single wireless link topology is considered, with data flowing in the wired server to wireless client direction, assumed to be the most common case [4]. As any TCP application can be viewed as a set of file transfers, it is tempting to assume that FTP throughput adequately characterizes wireless TCP performance. This generalization is however flawed. Real applications mostly rely on *short* data exchanges, thus TCP rarely reaches the throughput of FTP measurements. In addition, most TCP applications are either interactive or employ request/reply protocols, therefore data flows in *both* directions, and *each* data exchange must complete for the application to proceed, regardless of its direction and size.

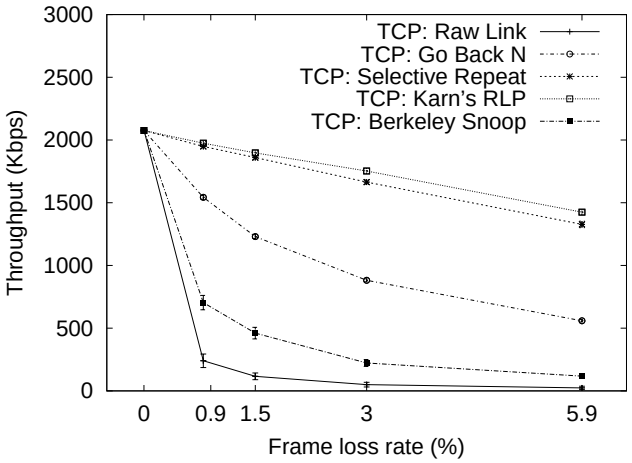

Fig. 5. WWW browsing throughput, one WLAN link

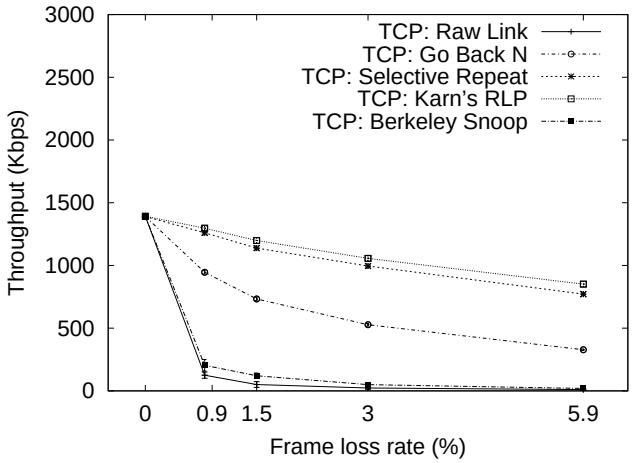

Fig. 6. WWW browsing throughput, two WLAN links

For a different perspective on TCP application performance we simulated *World Wide Web* (WWW) browsing over HTTP, the most popular Internet application [15]. In WWW 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 [15]. 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 for 2000 s (HSCSD) or 500 s (WLAN). We measured WWW browsing throughput, defined as the amount of *application* data transferred from the server to the client, including pages and embedded objects, divided by time taken. Client requests influence throughput by adding delays between these transfers. Measurements stop at the end of the last completed transaction within the simulation period.

Fig. 5 shows WWW browsing throughput with one WLAN link. Selective Repeat and Karn's RLP perform similarly, since their framing overhead is tiny for the large frames used. Interestingly, Karn's RLP beats Selective Repeat at higher loss rates. This is due to the short HTTP transfers: when the link becomes idle just after a packet loss, Selective Repeat does not receive any ACKs, resorting to timeouts to detect losses, while Karn's RLP sends *keepalives* when idle, thus always triggering NACKs. Berkeley Snoop is close to Raw Link, as it is unable to recover locally from lost client requests, while Go Back N lies between the other schemes. It is clear from these results that bidirectional recovery is essential for interactive applications, even if client requests represent only a small fraction of the total data transferred.

WWW browsing performance with two WLAN links is shown in Fig. 6. The only change from the previous case is that all schemes achieve lower throughput, due to the multiplicative loss effects of the two wireless links [22]. The exception is Berkeley Snoop which is even closer to Raw Link, as it is unable to retransmit both client requests in one direction and server replies in the other. It should be noted that HTTP and FTP throughput results are *not* comparable, since WWW browsing throughput incorporates client request delays. Both the one and two HSCSD link cases are similar to their WLAN counterparts.

Overall, our HTTP tests reveal that some TCP unaware link layer schemes offer considerable throughput improvements in all topologies, while the unidirectional recovery of Berkeley Snoop is unable to improve WWW browsing performance, even with a single wireless link. Our results again suggest that conflicting TCP and link layer retransmissions are less of a problem than resorting to TCP recovery. The most important insight is that the short bidirectional transfers of interactive applications clearly differentiate them from FTP transfers. Therefore, FTP throughput cannot capture the performance of interactive TCP applications. Similarly, unidirectional recovery is unable to improve the performance of interactive TCP applications.

D. Continuous media distribution

Applications which prefer UDP over TCP due to its simplicity, would work fine with the link layer schemes discussed above, but delay sensitive applications which use UDP in order to handle flow and congestion control themselves, are a poor match for fully reliable schemes. Thus, we tested UDP application performance with real-time continuous media distribution. We simulated a lecture where a speaker sends audio, and possibly video, to an audience including a wireless attendee. The speaker alternates between *talking* and *silent* states with exponential durations, averaging 1 s and 1.35 s, respectively [18]. Media are only transmitted when the speaker is talking. The speaker transmits media packets isochronously at a *Constant Bit Rate* (CBR) of 14.4 Kbps for HSCSD (speech), or 1 Mbps for WLAN (audio/video).

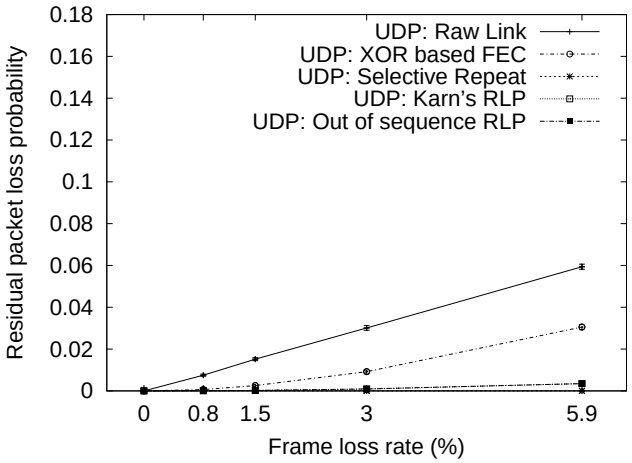

Fig. 7. Continuous media distribution loss, one WLAN link

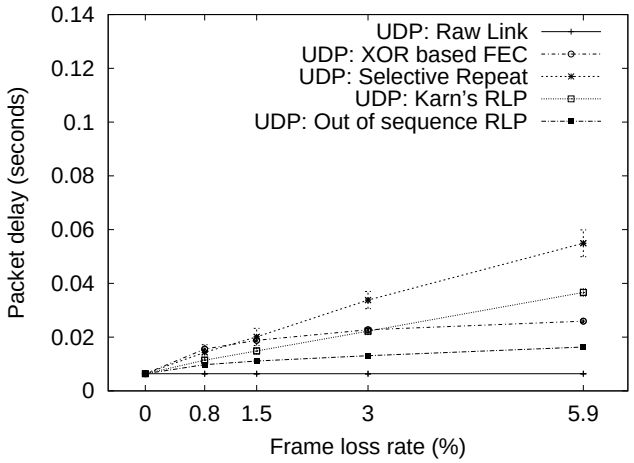

Fig. 8. Continuous media distribution delay, one WLAN link

We assume that the application employs a FEC scheme to tolerate some losses without retransmissions, thus enabling the sender to handle multiple receivers. We also assume that received packets are buffered until a *playback point* determined by human perception. This allows smooth playback despite delay variations, as long as most packets do not miss the playback point. Such measures are necessary to handle delays and losses due to congestion, but they are insufficient for wireless losses. To characterize application performance, we measured the *residual* loss rate at the receiver after link layer recovery. Since packets missing the playback point are dropped, this is coupled with a delay metric covering *most* packets. We used mean packet delay plus its standard deviation, to account for the variable delays introduced by each scheme. Each test lasted for 2000 s (HSCSD) or 500 s (WLAN).

Fig. 7 depicts residual losses with one WLAN link. Note that the transfer direction is unimportant, since all simulated schemes are symmetric. Raw Link exhibits the native loss rate, matching exactly the calculated loss rates. XOR based FEC, with 1 parity for 12 data frames, does not reduce losses in proportion to its overhead, e.g. the 3% native loss rate is reduced to 1% by adding 7.7% of overhead, as the parity frame is wasted both when no losses occur and when multiple losses occur. Selective Repeat always achieves full recovery, since it never stops retransmitting lost packets. Karn's RLP and our *Out of Sequence* (OOS) RLP, both with 1 retransmission per loss, perform identically, as their recovery mechanism is exactly the same. Both RLP variants considerably outperform the FEC scheme, while also introducing less overhead as only lost frames are retransmitted. Although limited recovery schemes do not completely eliminate losses, as long as they keep residual losses low enough for an error tolerant application, there is no need to resort to full recovery if it means higher delays. The results for the one HSCSD link case and both one and two WLAN link cases are similar.

Delay metrics with one WLAN link are shown in Fig. 8. Raw Link exhibits the native delay of the path. Selective Repeat suffers from high delay, since the sender may stall due to persistent losses and frames received correctly after a loss

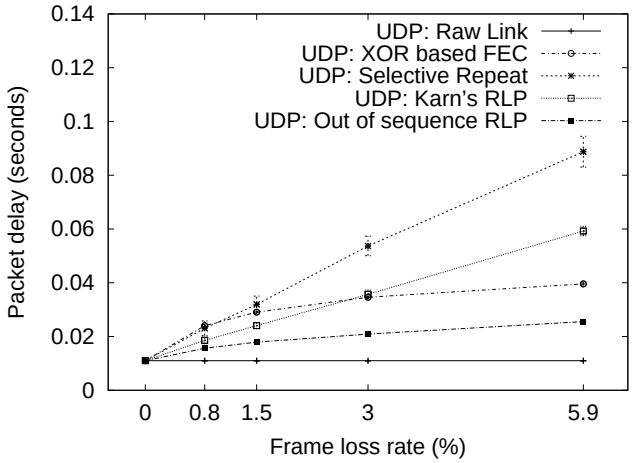

Fig. 9. Continuous media distribution delay, two WLAN links

are delayed at the receiver, in order to provide in sequence delivery. Karn's RLP is faster than Selective Repeat as it abandons recovery after a single failed retransmission, but the lowest delay among retransmission schemes is provided by OOS RLP which immediately releases received frames to higher layers. Interestingly, OOS RLP is faster than XOR based FEC. To understand why, consider how error recovery works. When a loss is detected, OOS RLP schemes sends a (short) NACK which triggers a retransmission, thus recovery takes one NACK plus one frame transmission delay. With XOR based FEC, 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, so retransmissions are *faster* than XOR based FEC with its large blocks.

Fig. 9 shows the delay metrics with two WLAN links. While all curves have the same shape as above, with a second WLAN link the delay metric for Karn's RLP increases between 61.7% and 62.4%, while for OOS RLP it increases between 56.7% and 61.4%. Therefore, the delay for Karn's RLP increases at a *faster* rate than that of OOS RLP. The reason is that with two

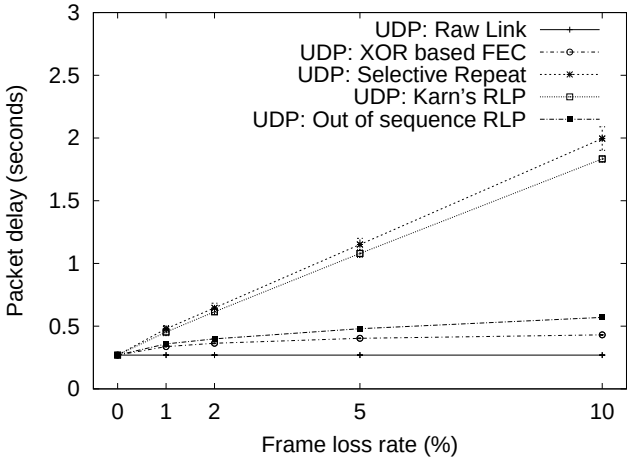

Fig. 10. Continuous media distribution delay, two HSCSD links

wireless links Karn's RLP may delay a frame *repeatedly* due to the loss of preceding frames, while OOS RLP delays only frames that must be actually retransmitted. Therefore, out of sequence delivery becomes increasingly appealing with multiple wireless links, even if the receiving application eventually resequences all packets in a playback buffer.

Delay metrics with two HSCSD links are presented in Fig. 10. Again, Selective Repeat exhibits the highest delay due to its full recovery policy, closely followed by Karn's RLP which suffers due to its in sequence delivery policy. 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 links. Note that in order to avoid high delays, the FEC scheme terminates blocks prematurely with a parity frame whenever the sender becomes silent, using a timeout of twice the regular packet interval. Delay results for the one HSCSD link case are similar.

Overall, our CBR tests reveal that the limited recovery RLP schemes adequately reduce residual losses for loss tolerant applications, unlike XOR based FEC with its relatively high overhead, or Selective Repeat whose full recovery is problematic for delay sensitive applications. While both RLP variants do not stall due to errors, OOS RLP further reduces delay over Karn's RLP for playback applications, with increased gains over multiple wireless links. Finally, OOS RLP is faster than block based FEC in low delay links with long frames and large blocks, thus the choice between retransmissions and FEC depends on the underlying wireless link.

IV. MULTI-SERVICE LINK LAYER ARCHITECTURE

The results presented above show that link layer error recovery can considerably improve Internet application performance over wireless links. This is encouraging, as link layer schemes can be locally deployed and optimized for the underlying link, transparently to the rest of the network. Unfortunately, different applications have different error recovery requirements, therefore multiple schemes must co-exist at the link layer. We have thus developed a *multi-service link layer* architecture supporting multiple such schemes in parallel.

To implement this concept, one issue is the number of *services*, i.e. link layer schemes, that will be provided. With one service per transport protocol we may bundle together dissimilar applications, leading to suboptimal performance. With one service per application we may overload the link layer in terms of both processing and memory resources. We therefore decided on one service per *application class*, i.e. per group of applications with similar requirements. All TCP applications belong to the same application class as TCP dictates their behavior. For UDP applications we expect diverse behaviors, one of which is the error tolerant but delay sensitive application class where continuous media distribution belongs.

Another issue is that traditional link layers provide a single service, thus adjacent layers expect the link layer to have unique entry and exit points. For this reason, our architecture provides a sublayer to distribute incoming packets to services and another sublayer to forward outgoing packets from all services to the next layer. These sublayers keep services unaware of their operation within a multi-service context, as they still have unique entry and exit points, therefore we can use existing link layer code to provide these services.

Fig. 11 outlines our architecture, showing data flow in one direction. Incoming packets from the network layer are passed to services based on their application class. Since higher layers are unaware of the multi-service concept, a classifier performs this task. A heuristic classification scheme is to check the IP *protocol* and the TCP/UDP *port* fields to detect the application in use [25]. When the *Differentiated Services* (DS) architecture is used, we can exploit the IP DS field [5] instead, which is visible even with IP security [11], unlike the protocol and port fields. Unmatched packets are passed to a default service providing access to the raw link.

Services may use any recovery mechanisms they desire. They may be optimized for the underlying link in terms of setting timeout values and window sizes. All services eventually pass their outgoing frames to the scheduler. The scheduler first tags each frame with a service number, encapsulating it into a multi-service frame, and this frame is then passed to the MAC sublayer for transmission. At the receiver, the MAC sublayer passes received frames to the demultiplexer which strips the service numbers and delivers them to the appropriate service. These services may eventually release their data to the multiplexer which simply forwards them to the network layer. Thus, the peer link layer services communicate over *virtual links*, transparently multiplexed over a single physical link. New services may be added by simply inserting modules at both ends of the link and extending classifier mappings.

Since services may arbitrarily inflate their data streams with error recovery overhead, if we transmit all frames in a FIFO manner the more aggressive services will grasp a larger share of the link. We have thus introduced a frame scheduler to ensure that the link is *impartially* shared. Impartiality means that each service should receive the same amount of bandwidth as in a single service link layer. To clarify this, assume that over some period the classifier allocates a_i bytes of incoming traffic to service i. With a single service, the data allocated to service i would consume a fraction $f_i = a_i / \sum_{j=1}^{n} a_j$ of

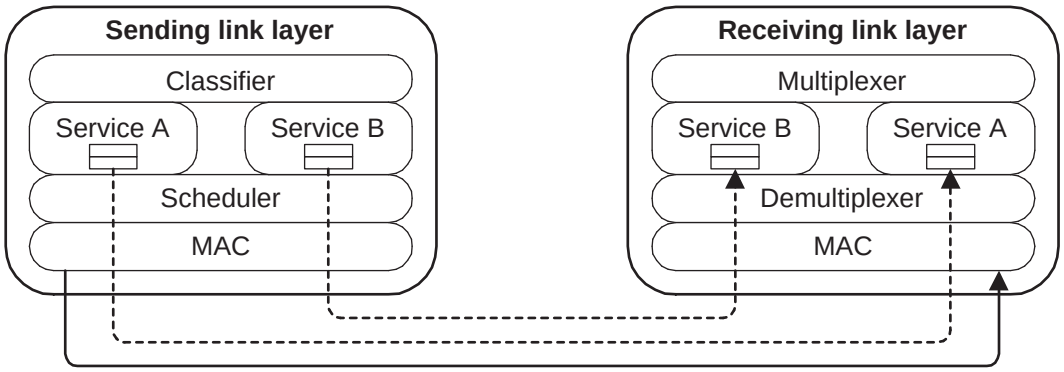

Fig. 11. Multi-service link layer architecture

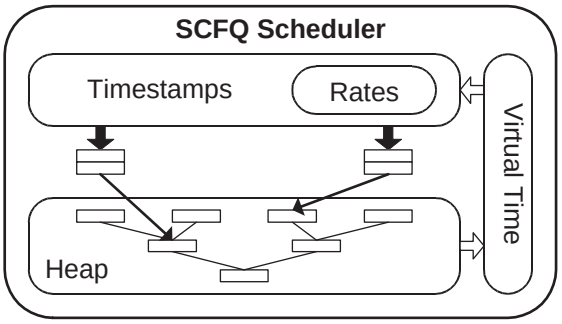

Fig. 12. Self clocked fair queueing frame scheduler

the total bandwidth. The scheduler is impartial if it allocates this exact fraction f_i of the total bandwidth to each service i. A service performing no error recovery will thus get the same data rate as with a single service. In contrast, a service performing error recovery will have to split its bandwidth between data and error recovery overhead. All applications belonging to the same application class employ the exact same service, therefore they are impartially treated with respect to each other.

Impartiality is achieved by a *Self Clocked Fair Queueing* (SCFQ) scheduler [9], shown in Fig. 12, which strictly enforces the desired bandwidth allocations. When some services are idle, their bandwidth is proportionately shared among the rest. The *rates* table holds the bandwidth fraction for each service. The scheduler maintains a *virtual time* variable, which is always equal to the *timestamp* of the last frame transmitted. Frames are kept in separate FIFO queues per service. To set the timestamp of an incoming frame we divide its size by its service rate and add it to the timestamp of the previous frame in its queue. If the queue is empty, we use the current virtual time. When the link is free, the frame with the *lowest* virtual time is dequeued for transmission and the virtual time is updated.

Since the time stamps in each queue are increasing, we only need to check the head of each queue to determine which frame has the lowest virtual time. The scheduler organizes the queues in a sorted *heap*, based on their heads, thus allowing immediate selection of the next frame to transmit. The heap must be sorted when a frame is removed for transmission, based on the next frame in the same queue. When a queue becomes empty, it is removed from the heap. When it becomes again non empty, it is added back to the heap and the heap is sorted. Therefore, each frame requires a few operations to calculate its timestamp and $O(\log_2 n)$ operations (for n services) to sort the heap when the frame leaves the scheduler for transmission and, possibly, when the frame enters the scheduler; the latter is needed when the frame is added to a previously empty queue.

A simple method to set the service rates is to measure the traffic allocated to each service by the classifier over a period of time, before any link layer overhead is introduced, thus dynamically calculating the f_i fractions. If a higher layer scheduler is used, the frames handed to the link layer will be already scheduled, so this method will preserve these decisions. This is consistent with our goal of protecting services from each other's error recovery overhead, leaving priority scheduling and admission control decisions to higher layers. Our architecture can also be extended by providing normalized performance metrics for each service, thus enabling higher layers to make intelligent decisions over wireless links [25].

V. MULTIPLE APPLICATION PERFORMANCE

A. Simulation setup

We evaluated our multi-service architecture by testing the TCP and UDP applications described above operating in parallel over different link layer schemes [23]. File transfer, WWW browsing and continuous media distribution executed over the same path, exactly as in single application tests, i.e. the TCP servers and UDP sender at one end and the TCP clients and UDP receiver at the other. All applications started together and the run ended when the file transfer completed. All metrics were finalized at the conclusion of the last completed WWW browsing transaction during the simulation period.

We implemented a multi-service link layer module for ns-2 supporting arbitrary numbers of services. We used the link layer schemes that performed best in single application tests to provide one TCP and one UDP service. The classifier assigned packets to services based on their IP DS fields [5] which were

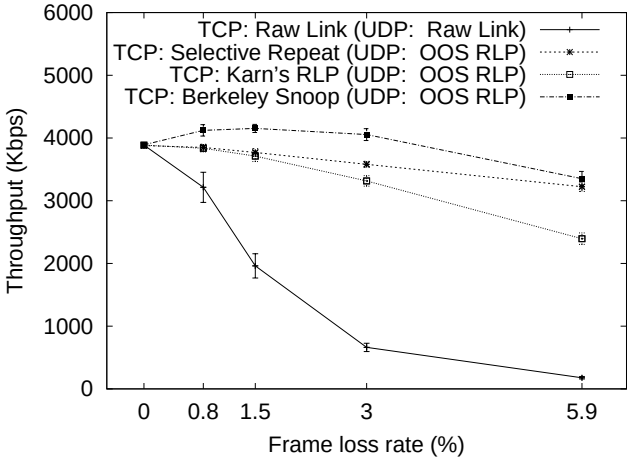

Fig. 13. File transfer throughput, one WLAN link

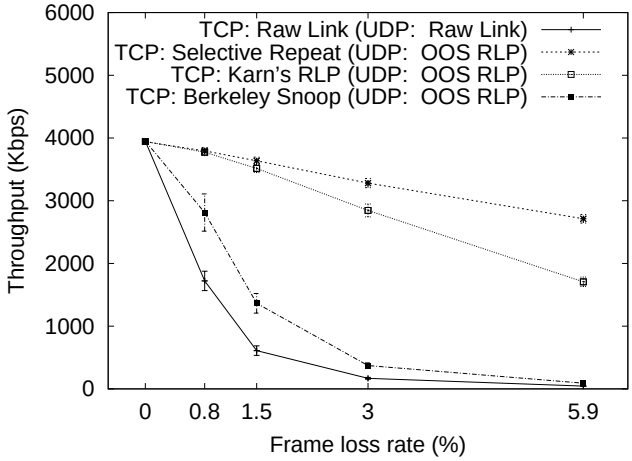

Fig. 14. File transfer throughput, two WLAN links

set by the applications. File transfer and WWW browsing traffic used the *same* service, allowing us to assess our approach with traffic differentiated by *application class*. After all, bandwidth sharing between competing TCP connections is handled by TCP congestion control mechanisms, thus there is no need to deal with it at the link layer. The rates for the SCFQ scheduler were set statically on each wireless link so that continuous media distribution was guaranteed its *peak* bandwidth, *before* adding link layer overhead. As the CBR application is not always active and the scheduler allocates all bandwidth to active applications, the bandwidth available for TCP was higher than what these rates imply.

B. File transfer

File transfer throughput results with one WLAN link are shown in Fig. 13. For each curve we show the scheme used for TCP (in parentheses, the scheme used for UDP). The results are similar to those of single application tests but with lower average throughput due to contention with the other applications. The main difference is that Berkeley Snoop performs better than Selective Repeat and Karn's RLP, with its throughput initially increasing at higher loss rates. As we will see, there is a corresponding performance degradation in WWW browsing with Berkeley Snoop, leaving more bandwidth for file transfer. Despite the low loss rates, all schemes considerably outperform Raw Link, indicating that while minor losses are easy to deal with at the link layer, they are disastrous for TCP.

Fig. 14 shows the corresponding results in the two WLAN link case. As in the previous case, Selective Repeat beats Karn's RLP by a margin increasing with higher loss rates. The main difference from above is that Berkeley Snoop provides only marginal gains over Raw Link. As in single application tests, this is due to the fact that it retransmits only in the wired to wireless direction, i.e. local recovery is performed over one wireless link in the path only.

Fig. 15 presents file transfer throughput with two HSCSD links. Karn's RLP lags behind Selective Repeat, while Berkeley Snoop is closer to Raw Link despite its lower link layer

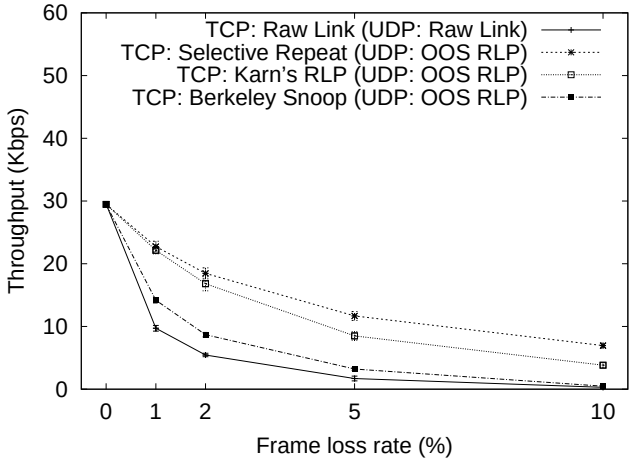

Fig. 15. File transfer throughput, two HSCSD links

overhead, again due to its inability to perform local recovery over both wireless links in the path. Results in the one HSCSD link case are similar to those of the one WLAN link case.

Overall, the FTP results in multiple application tests are similar to those of single application tests. Selective Repeat and Karn's RLP work well regardless of the underlying topology, while Berkeley Snoop works only in single wireless link topologies, with data transfers towards the wireless host. The only differences are the lower available bandwidth due to contention and the performance gains of Berkeley Snoop at the expense of WWW browsing.

C. World Wide Web browsing

Results for WWW browsing throughput in the one WLAN link case are shown in Fig 16. All curves are similar to those of single application tests, with reduced throughput due to contention. Selective Repeat and Karn's RLP perform well, as in the corresponding FTP tests. Berkeley Snoop with its unidirectional recovery fails to offer significant gains, as, even though most data flows in the wired to wireless direction, client requests in the reverse direction are critical for performance.

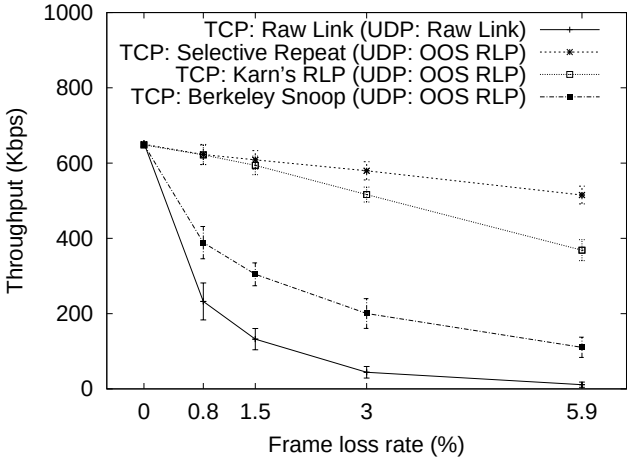

Fig. 16. WWW browsing throughput, one WLAN link

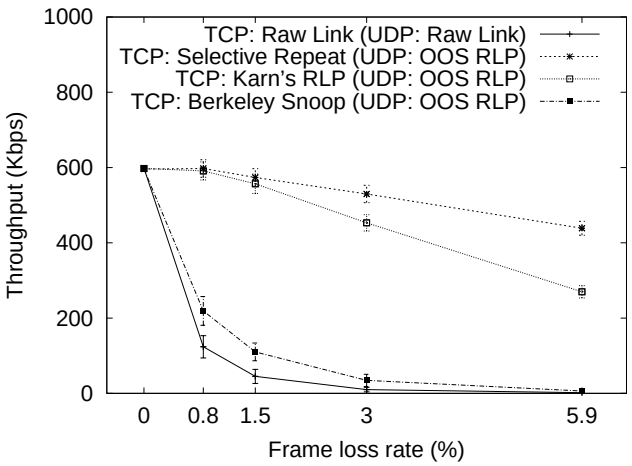

Fig. 17. WWW browsing throughput, two WLAN links

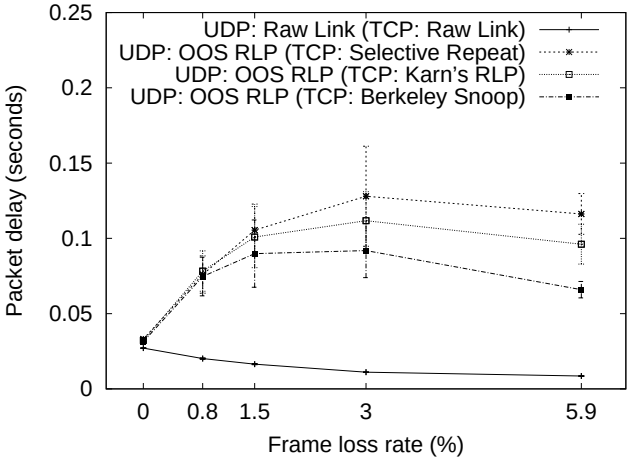

Fig. 18. Continuous media distribution delay, one WLAN link

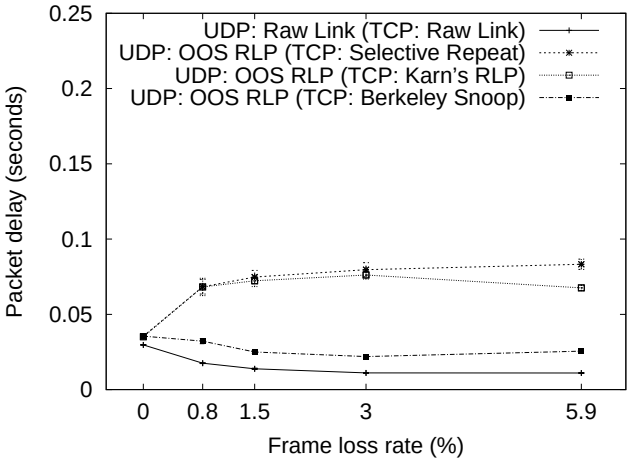

Fig. 19. Continuous media distribution delay, two WLAN links

The drop in WWW browsing throughput with Berkeley Snoop explains its increased FTP throughput in this scenario.

Fig 17 shows WWW browsing throughput with two WLAN links. The relative performance of most schemes is very similar to the one WLAN link case and to the corresponding FTP results, with Selective Repeat ahead of Karn's RLP. Berkeley Snoop is even closer to Raw Link, since in this scenario it cannot retransmit requests over one wireless link and replies over the other, thus repeatedly resorting to TCP error recovery. The results in both the one and two HSCSD link cases are similar to those of their WLAN counterparts.

Overall, the HTTP results in multiple application tests are again similar to those of single application tests. Selective Repeat and Karn's RLP offer considerable gains regardless of the underlying topology. The short bidirectional transfers of WWW browsing differentiate it from the unidirectional file transfers, showing that FTP cannot capture the performance of interactive applications. This is demonstrated by Berkeley Snoop which fails to improve the performance of WWW browsing in the same scenarios where it excels with file transfers.

D. Continuous media distribution

The residual loss metrics for continuous media distribution were the same as in single application tests, since the schemes used (Raw Link and OOS RLP) operate in exactly the same manner. The delay metrics with one WLAN link are shown in Fig. 18. For each curve we show the scheme used for UDP (in parentheses, the scheme used for TCP). With any non-preemptive scheduler, contention between TCP and UDP increases delay for both. The SCFQ scheduler however allocates sufficient bandwidth for the UDP application, thus it only suffers a modest delay increase. The differences between the three OOS RLP curves mirror the aggregate throughput offered by the corresponding TCP scheme. For example, with Raw Link delay *drops* with higher loss rates, as aggregate TCP throughput drops.

With two WLAN links the delay metrics, shown in Fig. 19, are similar to those with one WLAN link. The exception is Berkeley Snoop, since its performance for both TCP applications in this case is close to that of Raw Link, thus it only introduces a small additional delay for UDP. Finally, the delay metrics with two HSCSD links, shown in Fig 20, show all

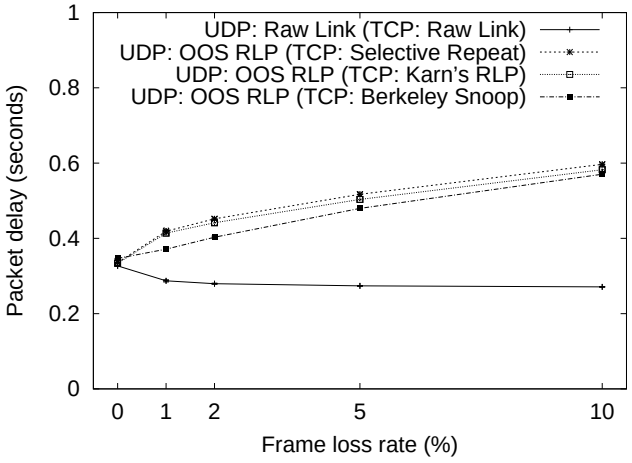

Fig. 20. Continuous media distribution delay, two HSCSD links

OOS RLP curves to be similar. The small differences between them are a consequence of the relatively small differences in aggregate performance with the corresponding TCP schemes. Results in the one HSCSD link case are very similar to those of the two HSCSD link case.

Overall, the CBR results in multiple application tests indicate that while loss rates are unchanged from the single application tests, the contention between TCP and UDP unavoidably leads to a delay increase for UDP traffic. Since part of the additional delay over Raw Link is due to the retransmissions of OOS RLP itself, the SCFQ frame scheduler actually manages to keep delay at acceptable levels for the UDP application.

VI. CONCLUSIONS

We have presented a comprehensive simulation study of diverse Internet applications, that is, file transfer and WWW browsing over TCP and continuous media distribution over UDP, using multiple wireless links and link layer schemes. This study extends our previous work [25] with new applications, wireless links, link layer protocols and topologies. Based on these results, we can draw a number of conclusions regarding the performance of Internet applications.

- Large file transfers are an inadequate model for interactive TCP applications, which form the majority of TCP traffic, as evidenced by the WWW browsing results.
- The TCP application tests suggest that retransmission conflicts between TCP and the link layer [7] are less of a problem than resorting to end-to-end TCP recovery.
- TCP unaware link layer schemes perform excellently for both file transfer and WWW browsing, unlike TCP aware schemes which fail with interactive applications.
- The same link layer schemes perform best for file transfer and WWW browsing, therefore unidirectional and bidirectional applications can be enhanced by a single scheme.
- The schemes that performed best for both TCP applications were different to those that performed best for the UDP continuous media distribution application.

Our work also verifies the following conclusions which were reported in earlier work [25].

- For the tight delay requirements of UDP based real-time applications, recovery can also be provided by limited retransmission schemes such as Karn's RLP or OOS RLP.
- For high speed and low delay links limited retransmissions may be *faster* than block based FEC, therefore we must always carefully consider the underlying link.
- Our out of sequence limited recovery scheme (OOS RLP) is a good match for playback applications, with more apparent gains in multiple wireless link paths.

We described a multi-service link layer architecture that aims to enhance the performance of diverse applications by supporting the simultaneous operation of multiple error recovery schemes. This allows the requirements of different application classes to be satisfied by the link layer. To evaluate our architecture, we repeated our tests with all applications executing in parallel. Based on these results we can draw a number of conclusions regarding our architecture.

- Application performance was similar to that of single application tests over the same link layer scheme, for all applications tested, despite the presence of multiple schemes.
- The TCP unaware link layer schemes improved both file transfer and WWW browsing, thus there is no need to provide separate services for different TCP applications.
- The additional delay for UDP due to contention with TCP was kept low by the scheduler, thus the SCFQ scheduler effectively protects services from each other.
- All performance enhancements were achieved by exactly the same schemes as in single application tests, thus existing link layer code can be reused.
- The packet classifier only used IP headers visible with IP security and the best schemes were transport layer unaware, thus performance can be improved transparently.

We therefore conclude that our multi-service link layer architecture can simultaneously provide performance enhancements for diverse Internet applications, based on link layer error recovery only and without requiring any changes to the rest of the Internet.

BIOGRAPHIES

- George Xylomenos is a member of the Mobile Multimedia Laboratory at the Athens University of Economics and Business, where he is a Lecturer of Computer Science. He received his Diploma in Informatics from the Athens University of Economics and Business, Greece, and M.S. and Ph.D. degrees in Computer Science from the Department of Computer Science and Engineering at the University of California, San Diego. His current research focuses on improving the performance of Internet applications over wireless links, providing Quality of Service over wireless and mobile networks and providing broadband services over next generation cellular networks. Email: `xgeorge@aueb.gr`. Web: `http://mm.aueb.gr/~xgeorge/`

- George C. Polyzos is leading the Mobile Multimedia Laboratory at the Athens University of Economics and Business, where he is a Professor of Computer Science. Previously, he was Professor of Computer Science and Engineering at the University of California, San Diego, where he was co-director of the Computer Systems Laboratory, member of the Steering Committee of the UCSD Center for Wireless Communications and Senior Fellow of the San Diego Supercomputer Center. He received his Dipl. in EE from the National Technical University of Athens, Greece and his M.A.Sc. in EE and Ph.D. in Computer Science from the University of Toronto. His current research interests include mobile multimedia communications, ubiquitous computing, wireless networks, Internet protocols, distributed multimedia, and performance analysis of computer and communications systems. Prof. Polyzos is on the editorial board of Wireless Communications and Mobile Computing and has been a guest editor for IEEE Personal Communications, ACM/Springer Mobile Networking, IEEE JSAC, and Computer Networks. He has been on the Program Committees of many conferences and workshops, as well as reviewer for NSF, the California MICRO program, the European Commission, and the Greek General Secretariat of Research and Technology and many scientific journals. Email: `polyzos@aueb.gr`. Web: `http://mm.aueb.gr/~polyzos/`

REFERENCES

[1] Multi service link layers for ns-2. http://www.mm.aueb.gr/~xgeorge/codes/codephen.htm[February 2004].
[2] UCB/LBNL/VINT network simulator - ns (version 2). http://www.isi.edu/nsnam/[February 2004].
[3] Badrinath BR, Bakre A, Imielinski T, Marantz R. Handling mobile clients: A case for indirect interaction. In *Proc. of the 4th Workshop on Workstation Operating Systems*. IEEE: Los Alamitos, 1993; 91–97.
[4] Balakrishnan H, Padmanabhan VN, Seshan S, Katz RH. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking* 1997; **5**(6): 756–769.
[5] Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. *RFC 2475* 1998.
[6] Brady PT. Evaluation of multireject, selective reject, and other protocol enhancements. *IEEE Transactions on Communications* 1987; **35**(6): 659–666.
[7] DeSimone A, Chuah MC, Yue OC. Throughput performance of transport-layer protocols over wireless LANs. In *Proc. of the IEEE GLOBECOM*. IEEE: New York, 1993; 542–549.
[8] Goff T, Moronski J, Phatak DS, Gupta V. Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. In *Proc. of the IEEE INFOCOM*. IEEE: Piscataway, 2000; 1537–1545.
[9] Golestani S. A self-clocked fair queueing scheme for broadband applications. In *Proc. of the IEEE INFOCOM '94*. IEEE: Los Alamitos, 1994; 636–646.
[10] Karn P. The Qualcomm CDMA digital cellular system. In *Proc. of the USENIX Mobile and Location-Independent Computing Symposium*, USENIX Association: Berkeley, 1993; 35–39.
[11] Kent S, Atkinson R. IP encapsulating security payload (ESP). *RFC 2406* 1998.
[12] Liu J, Singh S. ATCP: TCP for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications* 2001; **19**(7): 1300–1315.
[13] Ludwig R, Katz RH. The Eifel algorithm: making TCP robust against spurious retransmissions. *Computer Communications Review* 2000; **30**(1): 30–36.
[14] Ludwig R, Rathonyi B. Link layer enhancements for TCP/IP over GSM. In *Proc. of the IEEE INFOCOM*. IEEE: Piscataway, 1999; 415–422.
[15] Mah BA. An empirical model of HTTP network traffic. In *Proc. of the IEEE INFOCOM*. IEEE: Los Alamitos, 1997; 592–600.
[16] Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgment options. *RFC 2018* 1996.
[17] Nanda S, Eljak R, Doshi BT. A retransmission scheme for circuit-mode data on wireless links. *IEEE Journal on Selected Areas in Communications* 1994; **12**(8): 1338–1352.
[18] Nanda S, Goodman DJ, Timor U. Performance of PRMA: a packet voice protocol for cellular systems. *IEEE Transactions on Vehicular Technology* 1991; **40**(3): 584–598.
[19] Pang Q, Bigloo A, Leung VCM, Scholefield C. Performance evaluation of retransmission mechanisms in GPRS networks. In *Proc. of the IEEE WCNC*. IEEE: Piscataway, 2000; 1182–1186.
[20] Parsa C, Garcia-Luna-Aceves JJ. Improving TCP performance over wireless networks at the link layer. *Mobile Networks and Applications* 2000; **5**(1): 57–71.
[21] Stevens W. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. *RFC 2001* 1997.
[22] Xylomenos G, Polyzos GC. Internet protocol performance over networks with wireless links. *IEEE Network* 1999; **13**(5): 55–63.
[23] Xylomenos G, Polyzos GC. Multi-service link layer enhancements for the wireless Internet. In *Proc. of the 8th International Symposium on Computers and Communication*. IEEE: New York, 2003; 1147–1152.
[24] Xylomenos G, Polyzos GC. Wireless link layer enhancements for TCP and UDP applications. In *Proc. of the 17th International Parallel and Distributed Processing Symposium*. IEEE: New York, 2003; 225–232.
[25] Xylomenos G, Polyzos GC. Quality of service support over multi-service wireless Internet links. *Computer Networks* 2001; **37**(5): 601–615.