Link Layer Support for Quality of Service on Wireless Internet Links

George Xylomenos and George C. Polyzos
{xgeorge,polyzos@cs.ucsd.edu}
Center for Wireless Communications
and
Computer Systems Laboratory
Department of Computer Science & Engineering
University of California, San Diego
La Jolla, California, 92093-0114

***Abstract*— We have developed a novel link layer architecture that provides multiple Quality of Service points simultaneously over wireless Internet links. Our approach enhances the performance of diverse applications over error prone links. We discuss the performance problems of Internet protocols over wireless links, presenting as a case study our measurements on a wireless LAN, and argue that it is preferable to handle wireless impairments at the link layer. We present a simulation study of various link layer enhancements and their impact on TCP and UDP performance. Our results show that different approaches are preferable for each type of application. We thus propose a Multi Service Link Layer approach that supports multiple link layer mechanisms over a single link. Our scheme is transport protocol independent and customizable for the underlying wireless link technology. While our approach can be directly deployed on the existing Internet, it also provides support for future Quality of Service-aware protocols and applications. Our simulation results show that our approach improves uniformly the performance of both TCP and UDP applications.**

I. Introduction

During the past few years wireless communications have become very popular, with a multitude of competing technologies and service models available to the public. *Cellular Telephony* (CT) systems are evolving worldwide to fully digital technologies, while *Wireless Local Area Networks* (WLANs) that transparently link wireless hosts to the Internet are becoming cheaper and more interoperable. These systems, despite their differences, share characteristics that set them apart from both traditional (geostationary) satellite and wired links: low propagation delays compared to satellites and high error rates compared to wired links. Their error behavior varies, sometimes rapidly, due to factors such as interference, multipath fading, atmospheric conditions and possible user mobility, in a generally unpredictable manner. Due to economic and technological factors, typical wireless links are slower than wired ones, making them the most likely bottleneck of end-to-end paths.

The popularity of wireless systems has generated interest for their integration into the Internet. Although superficially easy, given the minimal requirements of Internet protocols, this task is complicated by hidden protocol dependencies on wired media. In addition, the unpredictable performance of wireless links hinders the evolution of the Internet to *Quality of Service* (QoS) provision. In Section II we discuss the causes and extent of these performance problems by reviewing previous work and presenting our own measurements. We argue that it is preferable to handle wireless errors at the link layer. In Section III we present a simulation study of various link layer enhancement mechanisms, showing that different solutions are preferable for each type of application. We thus describe in Section IV a novel *multi service link layer* architecture that provides multiple simultaneous services over a single link, so as to support diverse application requirements. We show how our approach can be deployed on the existing Internet or be integrated with future QoS oriented protocols. Finally, in Section V we present simulation results showing that our architecture can uniformly improve the performance of multiple applications.

II. Internet protocol performance over wireless links

A. The impact of wireless impairments

The Internet offers two main protocol choices at the transport layer. UDP does little more than provide direct access to IP, leaving higher layer protocols or applications to deal with the limitations of IP's best effort datagram delivery service. TCP offers instead a reliable byte stream service to its users, also taking care of flow and congestion control. TCP receivers generate cumulative acknowledgments for data received in sequence, while TCP senders retransmit data when they receive multiple (usually 3) duplicate acknowledgments. Since IP can reorder datagrams, fewer duplicate acknowledgments may not signify losses. TCP dynamically tracks the round trip delay on the end-to-end path and times out when acknowledgments are not received in time, retransmitting unacknowledged data. TCP assumes that all losses are due to congestion, thus loss detection also causes the sending rate to be reduced to a minimum and then increase gradually in order to probe the network's capacity [1].

Since wired links are extremely reliable, congestion is indeed the main cause of loss there. Congestion control requires end-to-end corrective actions, thus it is sensible to combine it with

loss recovery at the transport layer, simplifying the link layer and allowing applications that do not require absolute reliability to avoid it by using UDP. With wireless links in the picture however, frequent non congestion losses also occur, due to either wireless bit errors or communication pauses during handoffs. These cause the rate limiting congestion control mechanisms of TCP to be triggered repeatedly even for relatively low error rates, reducing throughput over the bottleneck wireless link [2]. End-to-end recovery also increases delay, a significant problem for interactive applications.

Performance measurements of a 900 MHz WLAN using UDP with 1400 byte packets over an 85 foot distance showed an average error rate of 1.55% with clustered losses [3]. TCP generally achieves lower throughputs than UDP, not only due to its additional overhead, but also because reverse traffic (acknowledgments) must share broadcast WLANs with forward (data) traffic. Delay and loss in broadcast media may also increase due to collisions. Measurements of TCP performance over a single hop path (a WLAN link with 2% packet loss) showed that TCP throughput dropped to only 47% of its value in the absence of losses [4]. CT links on the North American CDMA system exhibit 1-2% error rates over its 172 bit frames [5]. TCP/IP datagrams must be segmented into multiple such frames, causing IP datagram error rate to increase dramatically and TCP performance to drop accordingly. CT systems use interleaving techniques to randomize frame losses and avoid degraded voice quality. Random frame losses cause more IP datagrams to be lost than clustered losses, while interleaving also increases delay.

When multiple wireless links are included in a path, for example when two wireless hosts communicate over the Internet, errors accumulate, further reducing throughput. Wide area paths are more sensitive to losses since larger transmission windows must be maintained to fully utilize them and end-to-end recovery is slower there. Handoffs in cellular systems are another problem since TCP can timeout during communications pauses. An emerging concept is *hierarchical cellular* systems. These combine different technologies (satellites, CT, WLANs) to support heterogeneous overlapping cells. Roaming between dissimilar systems requires *vertical handoffs*, in addition to *horizontal handoffs* within a system. Vertical handoffs change long term end-to-end path characteristics.

B. Case study: A 2.4 GHz DSSS Wireless LAN

We have performed a detailed study of the Lucent 2.4 GHz WaveLAN, a 2 Mbps WLAN system that presents the same interface as an Ethernet LAN. We provide here a summary of our observations, while more details can be found in [6]. Our measurements extend previous work in many ways: we used hosts with varying processing power and different types of interfaces (ISA and PCMCIA); we studied bidirectional (TCP) in addition to unidirectional (UDP) communications to evaluate the impact of contention and collisions between data and acknowledgments; and we used Linux instead of a BSD derivative. We tested communication with single hop (WLAN) paths over which we had full control. We used a stock Linux kernel with drivers modified to report more detailed statistics.

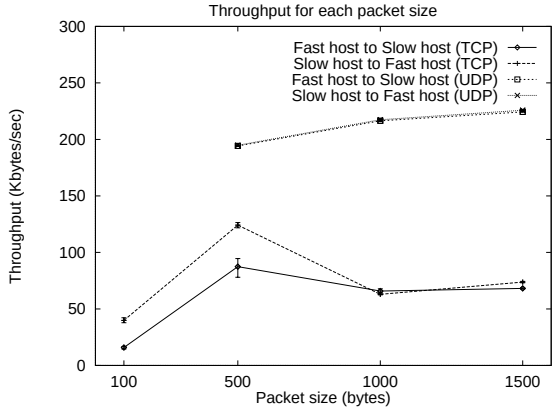

Fig. 1. UDP and TCP throughput (ISA/ISA)

Both ISA and PCMCIA interfaces, despite differences in their Ethernet controllers and radio modules, implement the same collision avoidance MAC scheme and are fully compatible. This replaces the collision detection MAC scheme of wired Ethernets because collision detection is costly in terms of bandwidth for radio links. After sensing an idle medium, the transmitter waits for a random number of slots and only if the medium is still silent does it transmit, to avoid excessive collisions. We used the `ttcp` benchmark which sends a number of packets using either UDP or TCP, modified to report loss statistics along with throughput. Each test consisted of sending 10,000 packets using either UDP or TCP, with different payloads (100, 500, 1000 and 1500 bytes). We repeated each test 5 times in both directions between each pair of hosts. We also used `tcpdump` to collect detailed packet traces.

We present here results from the baseline scenarios where the two communicating hosts are placed next to each other. The first scenario uses two ISA hosts, the second mixes one ISA and one PCMCIA host. For the ISA to ISA case, one of the hosts is considerably faster, causing data transfers towards the slower host to experience losses of one out of every roughly 1000 transmitted packets, depending on packet size. Figure 1 plots TCP and UDP throughput (mean, minimum and maximum across test repetitions) against packet size. UDP throughput peaks at 225 KBps (1.8 Mbps), higher than previously reported, due to increased host processing power. We do not show results for 100 byte packet UDP tests as the system dropped large bursts of those packets without sending them, due to protocol stack buffering limitations. TCP did not suffer from this problem due to its flow control.

The most interesting part is TCP performance with 1000 and 1500 byte packets where throughput drops to only about 30% of the corresponding UDP results. The cause can be seen in Figure 2 which shows the mean, minimum and maximum number of data and acknowledgment packets sent and received, with the faster host as the sender. The gaps between sent and received curves for both data and acknowledgments indicate losses that grow with packet size. These gaps are of the same magnitude and they are due to collisions undetected by the MAC layer. The packet traces show that the duplicate acknowledgments returned by the receiver after such a loss can also be lost to col-

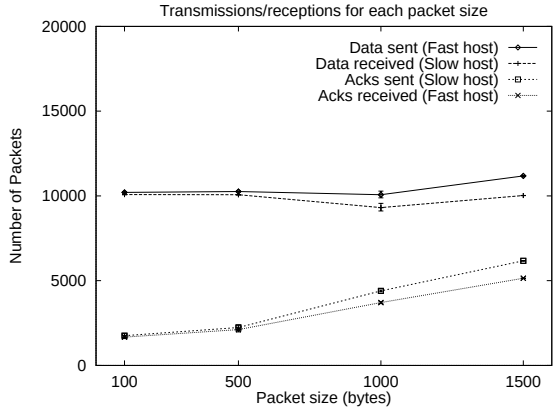

Fig. 2. TCP data and acknowledgments (ISA/ISA)

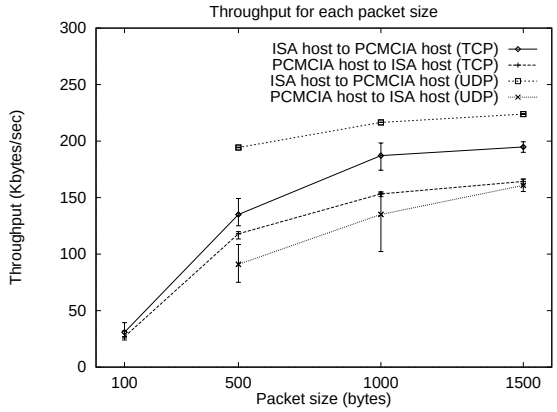

Fig. 3. UDP and TCP throughput (ISA/PCMCIA)

lisions. If the sender does not receive 3 duplicate acknowledgments however, it does not detect a loss. A similar problem occurs when the sender exhausts its transmission window before enough data packets are sent to trigger enough duplicate acknowledgments. In both cases, the sender must stall until a timeout occurs, which takes at least 200 ms in Linux and 500 ms in many BSD derived systems, leaving the link idle. Timeouts also lead to more drastic reductions in TCP's transmission rate than loss detection via duplicate acknowledgments [1].

In the ISA to PCMCIA case, the ISA host is considerably faster, again causing losses when sending towards the slower PCMCIA host. The PCMCIA host occasionally drops packets without sending them due to the single transmit buffer of PCMCIA cards (ISA cards have multiple buffers), which is easier to overrun. Figure 3 shows TCP and UDP throughput in both directions. TCP throughput is lower in the PCMCIA to ISA direction due to its less aggressive timing [3] and slower (single buffer) hardware implementation. In this scenario there are no excessive collisions due to the different timing of the two interfaces, leading to excellent TCP throughput. Interestingly, UDP is slower than TCP in the PCMCIA to ISA direction, despite TCP overhead and retransmissions. Packet traces indicate that occasionally the PCMCIA sender pauses for more than one second, exactly when packets are lost. Since, unlike TCP, UDP does not provide flow control, sender buffer overruns may occur causing interface resets. The resulting pauses cause peak UDP throughput to be only 160 KBps (1.28 Mbps) in this direction.

To summarize, asymmetries between processors and interfaces affect performance. Fast senders overrun slow receivers, causing packet losses, while PCMCIA cards lag behind ISA ones in transmission throughput. They are also prone to transmit buffer overruns and transmission pauses. The asymmetry between ISA and PCMCIA cards helps avoid the large number of undetected collisions that plague TCP performance in the ISA to ISA case. The impact of collisions on TCP is magnified by the loss of duplicate acknowledgments which leads to timeout initiated recovery. These shortcoming could be solved by MAC layer acknowledgments and retransmissions, thus enhancing TCP performance. UDP however does not face collision problems and interactive applications using it may prefer transmitting new data instead. The effects of timeout delays point out the importance of timer granularity on TCP. As long as adequate filtering is performed to avoid TCP instability, more accurate timers speed up recovery.

C. Link layer vs. transport layer enhancements

Since TCP is the most popular transport protocol on the Internet and its problems with wireless links are due to its own assumptions, modifying TCP has been a popular approach. The key is avoiding congestion control measures for non congestion losses. For mobility related losses, if signaling from lower layers is available, recovery may be initiated without a timeout [7]. For more frequent losses due to wireless errors, end-to-end recovery is too slow though. One way to speed it up is to *split* TCP connections where wired and wireless links interface. One instance of TCP executes over the wired part while another instance of TCP or another protocol executes over the wireless part [8], with software agents connecting the two. This allows wireless losses to be handled locally. Splitting end-to-end connections in this manner may violate transport layer semantics. Complex software agents are required to synchronize each transport connection. All TCP specific solutions are furthermore inappropriate for other protocols and since they operate at a high level they cannot exploit link specific optimizations.

The alternative to transport layer modifications is to improve the service offered on the link by providing error recovery at the link layer. Traditional link layers use retransmissions to provide full recovery, at the expense of variable delays. These may cause TCP to timeout in the process and trigger end-to-end loss recovery anyway [9]. In order to avoid such adverse interactions limited recovery may be provided, leaving further recovery to higher layers [5]. Another variation is to provide transport protocol specific optimizations at the link layer [10]. By exploiting transport layer information the link layer can avoid adverse interactions and economize on control overhead. This coupling however limits the applicability of such solutions to specific higher layers [2]. Link layer schemes generally have the advantage of working at the local level, with intimate knowledge of the wireless link and low round trip delays that allow fast recovery. They can be deployed locally, transparently to the rest of the Internet. A TCP study found link layer schemes to outperform transport layer ones over WLAN links [4]. For these reasons, we decided to focus our research on link layer enhancements.

III. Performance of link layer enhancements

A. Simulation environment and setup

Given the diversity of higher layer requirements, it is questionable whether a single link layer scheme is suitable for all applications. We thus undertook a simulation study to evaluate the performance of various types of mechanisms for different applications. We used the *Network Simulator* version 2 (ns-2) [11]. Ns-2 is oriented towards Internet simulations, containing very good implementations of many TCP variants. Its source code is freely available, thus researchers may extend the system by adding or modifying objects. The simulator is implemented in Object Tcl and C++, allowing objects to be prototyped in the Object Tcl interpreter and then compiled in C++ if desired. We fully exploited these capabilities, adding wireless error models, link layer schemes and applications.

We describe here performance measurements using CT wireless links with 14.4 Kbps of bandwidth (as in GSM) and a 100 ms frame delay (as in North American CDMA). The links employ fixed size 50 byte frames and suffer from independent frame losses at rates of 1%, 2%, 5% and 10%. The loss model and long delay are due to physical layer interleaving that aims to randomize frame losses [5]. Figure 4 depicts the two wireless link topology simulated which consists of two CT links joined by an Ethernet (10 Mbps, 1 ms), with the communicating parties being two wireless users at either end of the path. We also simulated a simpler topology consisting of a single CT link and an Ethernet, with the same parameters as above. In this topology the sender is at the wired end and the receiver at the wireless end of the path, simulating a wireless client receiving data from a wired server.

We tested both TCP and UDP applications with different requirements to examine the benefits of various link layer schemes for each type of application. For TCP, we simulated large file transfers using FTP over TCP Reno [1] with BSD style 500 ms granularity timers. In FTP tests the sender transmits 2 Mbytes of data to the receiver, with TCP handling flow, congestion and error control on an end-to-end basis. We ignored TCP/IP headers assuming that header compression is performed over the slow wireless links. Thus the total transfer consisted of 40,000 50 byte packets. We measured file transfer throughput, defined as the total size of the transfer (2 Mbytes), ignoring TCP and link layer overhead, divided by the total time taken to complete the transfer.

For UDP, we simulated one direction of an audio conference for 2000 seconds. We used a speech model with silence suppression: when the speaker is active, data are transmitted at a *constant bit rate* (CBR) of 9.6 Kbps (similar to North American CDMA); when the user is silent no data are sent. Talking and silent periods have exponential durations, averaging 1 sec and 1.35 sec, respectively [12]. The receiver buffers packets arriving at irregular intervals and plays them back isochronously, at a playback point determined by human perception. One metric for this application is the residual loss after link layer recovery. Since packets arriving after the playback point are unused, we are also interested in a delay bound for *most* packets. Thus, our delay metric was mean packet delay plus twice its standard deviation, which emphasizes the effect of variable delays. For all experiments we repeated each test 30 times with different random number generator seeds, calculating individual test metrics. We present here mean metric values among all tests, with error bars depicting the mean plus/minus one standard deviation.

For both TCP and UDP the *raw link* service is the performance baseline. *Selective repeat* is a window based full recovery mechanism. The sender buffers outgoing frames, and retransmits those that are not acknowledged before a timeout. The receiver acknowledges and releases frames received in sequence. Lost frames are negatively acknowledged so that they can be retransmitted without a timeout. The scheme tested provides multiple negative acknowledgments per frame to reduce timeouts [13]. If a frame is lost repeatedly, the sender eventually exhausts its window and stalls, thus delay is unbounded. An alternative is *Karn's RLP*, the *radio link protocol* of North American CDMA [5]. Negative acknowledgments are also used to trigger retransmissions, but if after a few (by default 3) retries the frame is not recovered, the receiver gives up, making the loss visible to higher layers. Limited recovery means that there is an upper bound on delay and the sender never stalls. These schemes are suitable for TCP since they only release frames in sequence. *Berkeley Snoop* is a TCP aware scheme [10]. It uses an agent at the base station (see Figure 4) that inspects TCP data and acknowledgments. Data sent to the wireless host are buffered, and if TCP acknowledgments indicate that a packet was lost, the packet is locally retransmitted. Snoop does not work for the reverse direction where TCP acknowledgments are returned from a remote host.

For the UDP conferencing application delay is more important than full recovery. *Forward error correction* (FEC) schemes provide limited recovery but fixed delay bounds by adding redundancy in the transmitted stream so as to allow the receiver to reconstruct the original data despite losses. The *XOR based FEC* scheme we tested transmits the original data frames unmodified, but every 8 frames a *parity* frame is constructed by XOR'ing the preceding data frames. If a single data frame is lost, it can be recovered by XOR'ing the remaining frames with the parity frame. FEC overhead is wasted when zero or more than one losses occur. This FEC scheme was selected for its simplicity. Even though not among the most powerful, it exhibits the basic behavior of FEC schemes. We also tested UDP over Karn's RLP with 1 retransmission, providing a fixed delay bound. Losses in this scheme cause subsequent frames that are correctly received to wait until the missing frame is received or abandoned. This in sequence delivery is critical for TCP but useless for applications with playback buffers. Thus, we also tested an *out of sequence* (OOS) RLP variant that reduces delay by releasing received frames immediately. When a path contains multiple wireless links, this prevents frames from being repeatedly delayed due to unrelated losses.

B. File transfer performance over TCP

Figure 5 shows file transfer throughput in the one wireless link topology (one Ethernet and one wireless CT link). TCP data flow from the wired to the wireless host with acknowledgments travelling in the reverse direction. Each curve depicts

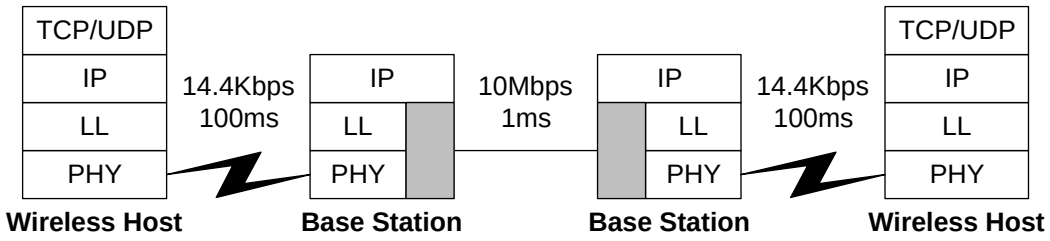

Fig. 4. Simulation topology.

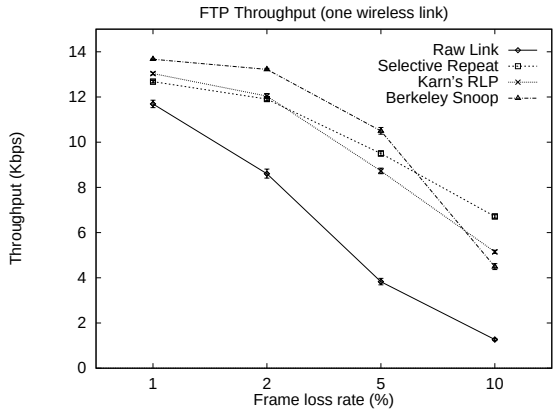

Fig. 5. File transfer throughput, one wireless link

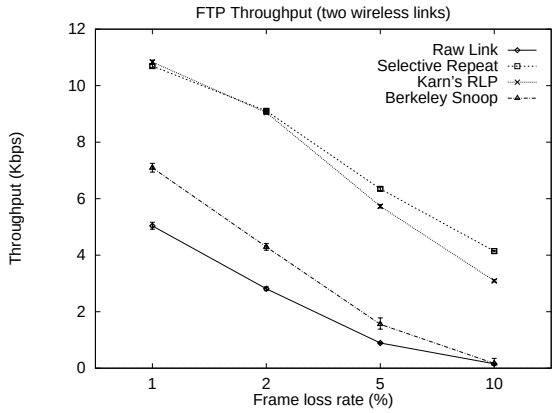

Fig. 6. File transfer throughput, two wireless links

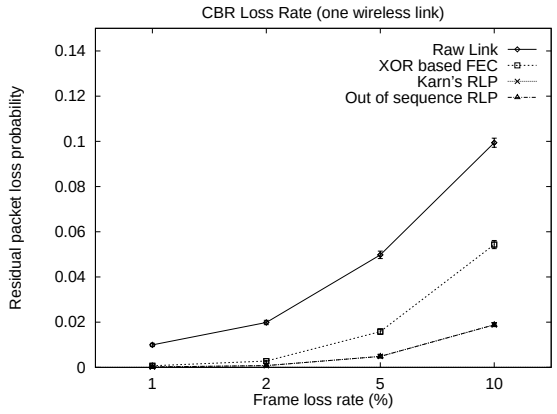

Fig. 7. Conferencing loss, one wireless link

average test throughput for a particular link layer scheme under varying error rates. The raw link curve shows performance without any enhancements, pointing out that losses disproportionately affect TCP performance: a 5% loss rate causes TCP to achieve less than 30% of the maximum throughput. The best performer in this topology is Berkeley Snoop for all but the highest loss rate. This is the most economical scheme in terms of overhead as it exploits existing TCP headers to detect losses. Karn's RLP uses one byte of overhead per frame, while selective repeat uses two bytes per frame. At higher error rates selective repeat is better, indicating that even though limited recovery avoids conflicting TCP and link layer retransmissions [9], it is better to recover without resorting to TCP. Since nearly all delay on this path is over the wireless link, the gains of link layer schemes are not due to local recovery but due to more robust error control. At the highest loss rate Snoop loses ground because it mimics TCP error recovery which is not very robust under harsh error conditions.

In the two wireless link topology, file transfer throughput, shown in Figure 6, reveals a different picture. Over the raw link TCP achieves about half of its throughput in the previous topology. Karn's RLP and selective repeat provide large performance improvements even under low loss rates, with selective repeat becoming more attractive under harsher error conditions. Berkeley Snoop however offers only minor improvements over the native service. The reason is its inability to improve performance in both directions over each wireless link. In this topology data cross the first wireless link in the wireless to base direction and the second in the base to wireless direction. Snoop improves only the latter, hence losses in the former are visible to TCP. The same problem occurs in wireless to wired host transfers or with interactive applications like WWW browsing that exchange data in both directions. This problem cannot be avoided by placing an agent at the wireless host, as acknowledgments in this direction are returned much later by a remote host, hence negating the advantages of local recovery. The other link layer schemes do not face this problem as they use local acknowledgments for error detection.

C. Conferencing performance over UDP

For the conferencing application the first metric of interest is residual loss, shown in Figure 7 for the one wireless link topology. Instead of selective repeat and Berkeley Snoop which are inappropriate for delay sensitive traffic, we used XOR based

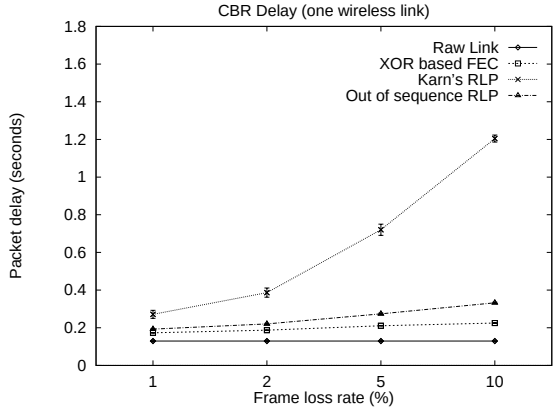

Fig. 8. Conferencing delay (mean + 2 × std. deviation), one wireless link

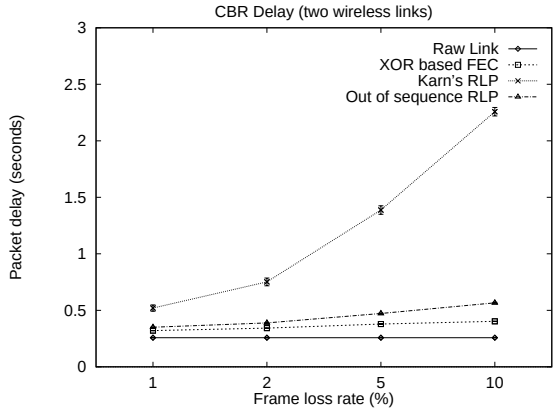

Fig. 9. Conferencing delay (mean + 2 × std. deviation), two wireless links

FEC and out of sequence (OOS) RLP, with a one retransmission limit for both RLP variants. OOS RLP was inappropriate for TCP as it did not deliver data in sequence. The raw link curve shows the native loss rate of the wireless link. The inefficiency of the FEC scheme is clear at the 10% loss rate: despite its 12.5% overhead (1 parity frame per 8 data frames) it only reduces the error rate to 6%. Both RLP variants keep loss rates below 2% in all cases, with identical performance throughout the error rate range, as expected. Loss rates for the two wireless link topology (not shown) are essentially twice those of Figure 7. These results confirm that both RLP schemes provide more efficient error recovery than XOR based FEC.

Figure 8 shows the delay metric for the one wireless link scenario, with the raw link curve depicting the native delay. The worst performer is Karn's RLP: as loss rates grow, delay increases dramatically due to the large number of retransmissions that inflate delay variability. Lost packets also delay subsequently received ones due to in sequence delivery. Out of sequence RLP on the other hand never delays received packets, hence it is very close to XOR based FEC, the best performing scheme. RLP recovery delay is determined by round trip delay, while FEC delay is determined by block size (and one way propagation delay).

With two wireless links the delay metrics, shown in Figure 9, are similar: XOR based FEC depicts the lowest delay among link layer enhancement schemes, closely followed by out of sequence RLP, with Karn's RLP exhibiting very large delays at higher error rates. While Karn's RLP increases its delay compared to the one wireless link topology by 90%, OOS RLP increases it by 70%, thus the gap between them widens. This confirms that out of sequence delivery is more beneficial with multiple wireless links since packets are not repeatedly delayed due to unrelated losses. This is a very important point for multi hop wireless networks.

IV. The Multi Service Link Layer approach

A. The case for multiple link layer services

Our results show that link layer schemes can significantly improve Internet performance over wireless links. Link layer mechanisms provide a *local* solution to a problem that is inherently dependent on and isolated to individual links. Unlike higher layer schemes, their scope is limited to the endpoints of the wireless link. Local schemes can be optimized for each type of link since hardware details are accessible at the link layer. Error recovery overhead is minimized, since no retransmitted or parity frames travel beyond the wireless link. Delays are also minimized since awareness of link characteristics can be used to set tight timeout intervals.

Our results however also indicate that different applications prefer fundamentally different enhancement mechanisms. TCP based applications are relatively insensitive to delay but very sensitive to losses, favoring full recovery schemes. Real time UDP based applications are relatively error tolerant but very delay sensitive, favoring limited recovery schemes. Future applications may exhibit other unforeseen requirements. As the Internet evolves towards providing multiple Quality of Service (QoS) points, as in the *Differentiated Services* architecture [14], the limitations of link layer schemes providing a single service will become more apparent. In order to provide multiple QoS points and support applications with diverse requirements we are thus proposing a *multi service link layer* architecture offering multiple simultaneous services over each wireless link. The architecture is easy to extend with new services catering to future requirements. Providing multiple services at the link layer is probably the only way to locally handle wireless impairments in a protocol independent manner.

B. Internal design

In order to provide multiple simultaneous services at the link layer we need to handle a number of design issues. Services must be isolated from each other so as to simplify their implementation, while at the same time they must share link resources in an orderly manner. The multi service aspect must be transparent to the existing Internet which assumes that only a single service is offered, but visible to and easy to integrate with future QoS based higher layers and applications. Figure 10 depicts the internal design of our multi service link layer architecture. A single entry point is provided for compatibility with IP, with an internal *packet classifier* distributing incoming packets to appropriate services. The classifier examines a number of IP and TCP/UDP header fields in order to map higher layer requirements to available services. QoS based higher layers can explicitly set up these mappings, or the scheduler may

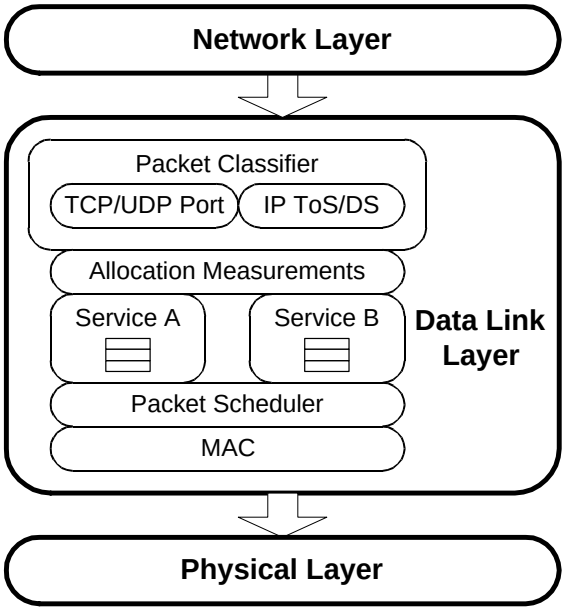

Fig. 10. Multi service link layer architecture.

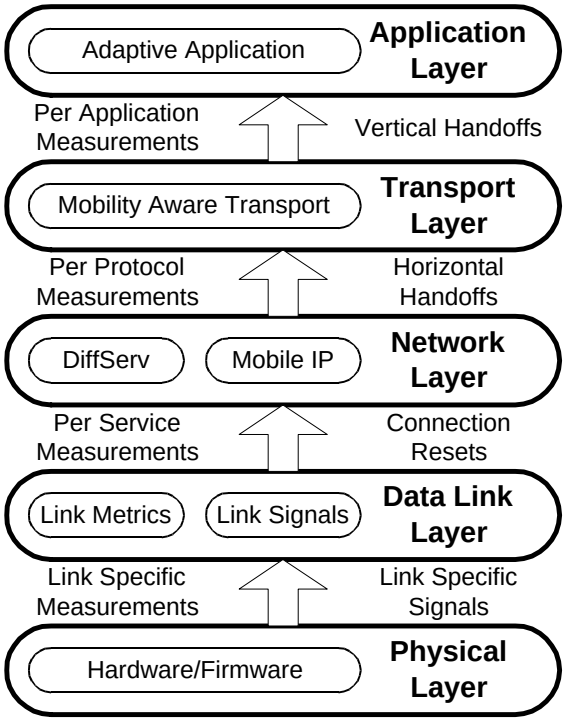

Fig. 11. Feedback propagation through the protocol stack.

heuristically map packets to services to ease integration with the existing Internet. Packets that are not explicitly matched to customized services are handled by the raw link service.

Higher layers expect the link layer to allocate transmission bandwidth to packets in the proportions presented to it by the network layer. This is especially important for higher layers that provide QoS guarantees based on packet scheduling. Preserving bandwidth allocations is trivial in traditional link layers, but more complicated with multiple services. If a service provides error recovery, whether using retransmissions or data encoding, the overhead it generates inflates its data stream relative to other services. In order to allow each service to use any mechanism desired but prevent it from adversely interacting the rest, we measure service bandwidth allocations at the entry to the link layer and enforce the same allocations over the physical link. The *allocation measurement* module tracks the fraction of data allocated to each service by the classifier over a time interval, before services inflate their data streams. These shares are enforced by the *packet scheduler* over the next time interval.

We decided to use a *self clocked fair queueing* (SCFQ) scheduler as it is very efficient to operate [15]. SCFQ schedulers strictly enforce the desired bandwidth allocations when the link is loaded. When some services are inactive, their bandwidth is shared among the active ones, hence the link is never left idle when there are data to send. Note that the goal of the link layer scheduler is *not* end-to-end QoS provision but the preservation of higher layer scheduling decisions despite arbitrary service overheads. This design isolates services from each other, since each can operate as if it was a conventional link layer protocol with a single peer at the other end of the link, using any recovery mechanisms desired and taking advantage of any possible link specific optimizations.

Incoming packets are passed to services by the classifier using a generic call. Services pass packets to the scheduler using per service queues. The scheduler decides which packet to send next based on the measured bandwidth allocations, labels it with a service number, and hands it to the MAC layer for transmission. At the receiver, incoming packets from the MAC layer are demultiplexed based on their service label and passed to the appropriate service, which may eventually pass them to the network layer. Hence, the classifier and scheduler are independent of the actual services implemented, and therefore reusable. Adding or removing services only requires modifying the mapping tables of the classifier, making the scheme very easy to extend.

C. External interface

For existing Internet protocols, the interface presented by our multi service link layer scheme is identical to that of traditional link layers, with single entry and exit points. Application specific enhancements take place transparently to the rest of the stack. QoS aware higher layers on the other hand need information about the services offered so as to select the best one for each type of traffic. Since each wireless link has different characteristics, rather than attempting to standardize the offered services across the Internet, it is preferable to allow dynamic discovery of their characteristics. Thus, each service in our scheme exports dynamically updated device independent metrics of its recent performance. Figure 11 depicts this upward flow of information through the protocol stack.

The physical layer may provide hardware specific performance feedback to the link layer, which combines it with its own measurements. Each service separately tracks and reports its performance metrics, i.e. residual loss, delay and throughput. Throughput is normalized to reflect service performance as if it were using the link exclusively, to make it independent

of past bandwidth allocations. Metrics are expressed in link independent units such as bits and seconds. Besides making dynamic service selection possible, these metrics allow higher layers to estimate end-to-end path performance before and during a session [16]. The metrics may be used directly or be modified by intermediate layers, for example to account for their own error recovery mechanisms.

The link layer cannot handle mobility problems due to handoffs, as one of the two endpoints changes after a handoff. Due to its intimate knowledge of the underlying hardware it can however help higher layers deal with mobility. Notifications, in the form of upcalls to higher layers, can inform interested parties of events such as disconnections and resets, that may indicate the beginning and end of handoffs. These events are detected by link layer state changes or physical layer signals and are propagated in a standardized manner through the protocol stack. They can be filtered by higher layers into authoritative mobility notifications (see Figure 11), thus enabling protocols and applications to adapt their mechanisms accordingly. For example, a disconnect notification and a connect notification from two different link endpoints indicate to the network layer that a handoff occurred.

D. Integration with the Internet

For our link layer architecture to succeed, it must seamlessly integrate with both existing and future (QoS aware) Internet protocols. Each service provided employs a mix of mechanisms optimized for the underlying medium, possibly adapting to current link conditions. Services are meant to satisfy generic requirements rather than fit the specific details of TCP or UDP applications. In the absence of any explicit instructions, application requirements may be inferred by examining IP and TCP/UDP headers. The classifier may use fields such as protocol type, TCP/UDP port number for applications using well known ports, or the IP *Type of Service* (ToS) field to heuristically map incoming traffic to link layer services. Unrecognized traffic is passed to the raw link (or other default) service. New services may be added as needed to expand the range of supported traffic classes or handle future requirements.

For QoS based higher layers, our link layer enhances the offered service without interfering with higher layer decisions. When packet scheduling is used to control bandwidth sharing during congestion, as in *Class Based Queuing* (CBQ) [17], our link layer preserves such allocations. The overhead introduced by each service does not violate higher layer scheduling since our scheduler allocates bandwidth based on the incoming data stream before it is inflated by each service. Higher layers aware of the multiple services offered may create enhanced end-to-end services using RSVP for setup and negotiations [18]. The performance metrics exported by each service can be used to select and characterize services. When *Differentiated Services* are used to provide various QoS grades to different traffic classes, the same IP DS field may be used to map packets to services at both levels [14].

Link layer feedback can be further refined as it flows upwards through the protocol stack, as shown in Figure 11. Mobile IP can rapidly detect mobility using link layer notifications, in turn issuing separate notifications for horizontal and vertical

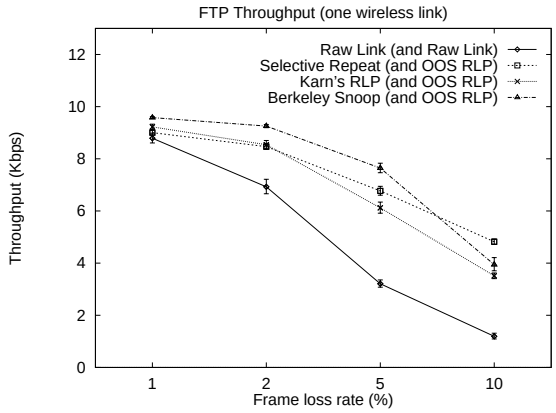

Fig. 12. File transfer throughput, one wireless link

handoffs. Horizontal handoffs can be dealt with at the transport layer, for example by freezing TCP timers until the handoff completes. Vertical handoffs change long term path behavior, so they are propagated to applications. A video application for example could adapt by switching to a different resolution. The services available on the new link can be discovered by looking at the exported metrics. End-to- end services can then be renegotiated by signaling protocols such as RSVP. Higher layer packet scheduling schemes such as CBQ may also use the exported metrics to predict the outcome of their scheduling decisions.

Deployment of the proposed multi service link layer architecture on the existing Internet is eased by its emulation of a single service link layer. Both endpoints of each wireless link are usually under the control of a single administrative entity, thus both sides can upgrade their link layer software at the same time transparently to the rest of the Internet. Generic services, such as those discussed in Section III, may be provided at first to efficiently support all TCP and some UDP applications, with further services added as future needs arise. This is attractive to equipment vendors that could deploy our scheme so as to gain a competitive advantage on the market.

V. Multi Service Link Layer performance

To evaluate our architecture we added a multi service link layer object to the ns-2 simulator (see Section III). Our implementation includes a heuristic classifier that maps packets to services based on their protocol (TCP or UDP) and a SCFQ scheduler, both capable of handling an arbitrary number of services. Services are implemented using any of the link layer enhancement schemes described previously. We wanted to determine whether our approach is capable of simultaneously enhancing the performance of multiple applications by the same factor as when executing each application in isolation over the same link layer mechanism, adjusted for the reduced bandwidth due to link sharing. We thus simulated simultaneous file transfer over TCP and conferencing over UDP using the same CT links, topologies and application parameters as in Section III. Both applications start simultaneously over the same path and the simulation ends when FTP completes sending 2 Mbytes of data.

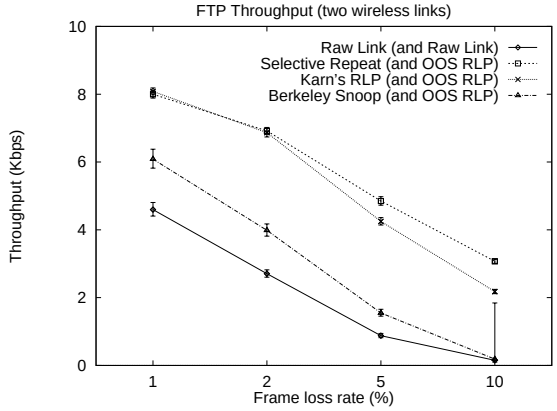

Fig. 13. File transfer throughput, two wireless links

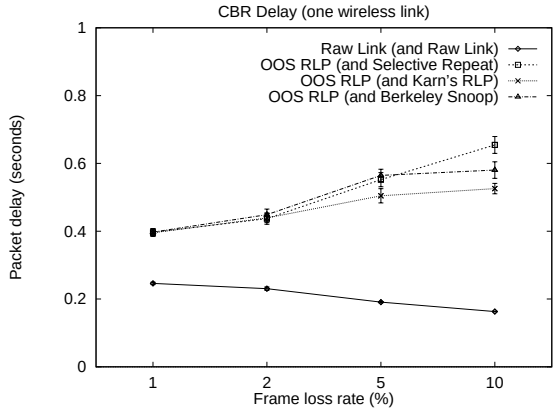

Fig. 14. Conferencing delay (mean + 2 × std. deviation), one wireless link

Each wireless endpoint employs a multi service link layer module with one TCP and one UDP service. We statically allocated 9.6 Kbps of bandwidth (out of 14.4 Kbps) to the UDP service, satisfying the peak CBR requirements of the conferencing application. Since the UDP sender is active 42.5% of the time, the average bandwidth available to TCP is 10.3 Kbps. To establish a performance baseline we used the raw link service for both UDP and TCP. Enhanced performance was provided by out of sequence (OOS) RLP for UDP, coupled with selective repeat, Karn's RLP or Berkeley Snoop for TCP. We show average results from 30 test repetitions.

Figure 12 shows file transfer throughput in multi service tests using the one wireless link topology, while Figure 13 shows the same metric with the two wireless link topology. The service used for UDP is shown in parentheses. The shape and relative position of all throughput curves are nearly identical to those shown in Figures 5 and 6. While performance is reduced for all schemes due to competing UDP traffic, the factor of improvement over the raw link service is virtually the same as in single application tests using the same link layer mechanism. Thus, TCP performance over multi service links is improved in exactly the same manner as with a single service.

For the UDP service, residual loss results (not shown) are exactly the same as in single service tests (see Figure 7). Figure 14 shows the delay metric for conferencing in multi service tests with one wireless link, while Figure 15 shows the same metric

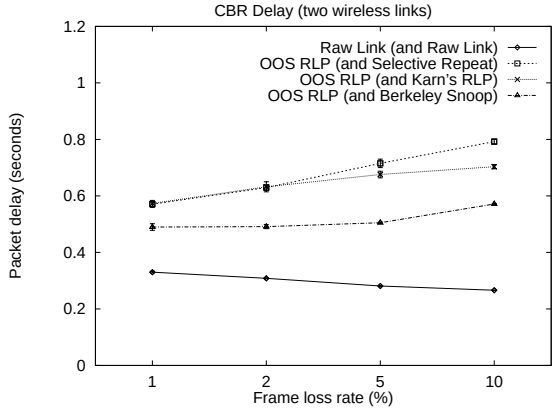

Fig. 15. Conferencing delay (mean + 2 × std. deviation), two wireless links

with two wireless links. The service used for TCP is shown in parentheses. Delay is inflated compared to the raw link scenario due to OOS RLP recovery delay (see Figures 8 and 9) and contention with TCP traffic. When TCP uses the raw link service its performance drops with higher error rates, hence UDP delay also drops. When enhanced TCP services are used, TCP performance improves, hence UDP delay increases. Note however that delay increases in proportion to the improvement in TCP performance and *not* to error recovery overhead. Thus, the scheduler manages to protect UDP delay performance from TCP service overhead.

VI. Conclusion

We outlined the problems faced by Internet protocols over wireless links and argued in favor of link layer enhancements. We presented simulations showing that diverse applications are best served by fundamentally different link layer schemes. We thus proposed a multi service link layer architecture that simultaneously improves the performance of multiple types of applications by combining arbitrary link layer mechanisms over a single link. Our link layer exports information about its services so as to allow enhanced end-to-end services to be created. Services are isolated from each other to ease programming. Our architecture can be transparently integrated with the existing Internet. Our simulations show that our approach offers virtually the same improvements to each stream sharing the link as those of single application tests, the best we could have hoped for while preserving fairness.

References

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

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

[3] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proc. of the Winter Simulation Conference*, 1996.

[4] 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*, August 1996, pp. 256–267.

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

[6] G. Xylomenos and G. C. Polyzos, "TCP & UDP performance over a wireless LAN," in *Proc. of the IEEE INFOCOM '99*, March 1999, pp. 439–446.
[7] R. Cáceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 850–857, June 1995.
[8] 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*, October 1993, pp. 91–97.
[9] 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*, December 1993, pp. 542–549.
[10] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995.
[11] "UCB/LBNL/VINT Network Simulator - ns (version 2)," available at http://www-mash.cs.berkeley.edu/ns.
[12] 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.
[13] 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.
[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," RFC 2475, December 1998.
[15] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proc. of the IEEE INFOCOM '94*, June 1994, pp. 636–646.
[16] S. Shenker and L. Breslau, "Two issues in reservation establishment," in *Proc. of the ACM SIGCOMM '95*, October 1995, pp. 14–26.
[17] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, August 1995.
[18] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, vol. 7, no. 5, pp. 8–18, September 1993.