Quality of Service Support over Multi-Service Wireless Internet Links

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*—Internet application performance over wireless links is disappointing, since wireless impairments adversely affect higher protocol layers. In order to address these problems without global protocol modifications, we examine link layer enhancement schemes. Simulations show that different schemes work best for different applications. We have thus developed a multi-service link layer architecture that simultaneously enhances the performance of diverse applications by supporting multiple link mechanisms concurrently. Simulations show that our approach dramatically improves performance. We present various ways of embedding this architecture into the Internet, thus allowing applications to select themselves the appropriate trade-off between throughput, loss and delay.**

I. Introduction

The Internet usually adopts new communications technologies quickly, largely due to the physical layer independence of the *Internet Protocol* (IP), which offers a standardized interface to higher layers regardless of the underlying link. The explosive growth of the Internet in the past few years is only paralleled by the growth in wireless personal communications. Digital *Cellular Communications* (CC) are spreading worldwide and evolving towards third generation systems, while *Wireless Local Area Networks* (WLANs) are finally starting to conform to international standards. Due to physical and economic constraints however, wireless links lag behind wired ones in performance. The popularity of such systems has motivated considerable work on their integration into the Internet. Even though satellite links have long been a part of the Internet, higher layer protocols and applications commonly make assumptions about link performance that cannot be met even by the terrestrial CC and WLAN links. Thus, although providing IP services over wireless links is easy, the resulting performance is disappointing. As the Internet evolves towards the provision of *Quality of Service* (QoS) guarantees, in order to support applications such as real-time multimedia communications, improving wireless link performance will only become more critical.

This article presents an architecture that improves the performance of diverse Internet applications while extending QoS support over wireless links. In Sect. II we outline the problem and review previous approaches, arguing for a link layer solution. In Sect. III we present simulations showing that different applications favor different link enhancement schemes. In Sect. IV we present a *multi-service link layer* architecture that simultaneously enhances the performance of diverse applications by supporting multiple link mechanisms in parallel. In Sect. V we present simulations showing that with our architecture each application achieves similar gains as when it operates by itself over its preferred link mechanism. The remainder of the article discusses how our architecture fits into the context of Internet QoS approaches: Sect. VI covers the existing best-effort service, Sect. VII the Differentiated Services architecture, and Sect. VIII a dynamic service discovery architecture.

Corresponding author. Address: Prof. George C. Polyzos, Dept. of Informatics, Athens University of Economics and Business, Patision 76, Athens 104 34, Greece. Tel: +30-1-8203650. Fax: +30-1-8203325. E-mail: polyzos@aueb.gr

II. Background

IP provides an unreliable packet delivery service between any two hosts: packets may be lost, reordered or duplicated. Applications can use the *User Datagram Protocol* (UDP) for direct access to this service. Some applications employ UDP assuming that the network is reliable enough, for example file sharing via the Network File System over LANs. Delay sensitive applications may also use UDP, adding their own custom error recovery mechanisms. For example, real-time conferencing applications may add redundancy to their data to tolerate some errors without the need for end-to-end retransmissions.

Most applications however require complete reliability, therefore they employ the *Transmission Control Protocol* (TCP), which provides a reliable byte stream service. TCP breaks the application data stream into segments which are reassembled by the receiver. The receiver returns cumulative acknowledgments (ACKs) for those segments received in sequence, with duplicate acknowledgments (DUPACKs) for reordered ones. Since packets may be reordered by IP, even though a lost data segment leads to a DUPACK when the next segment arrives, the sender retransmits the (apparently lost) next segment in sequence only after multiple (usually 3) DUPACKs are returned. The sender tracks the round trip delay of the connection, so that if an ACK for a segment does not arrive on time, the segment is retransmitted [2].

The error rate of wired links is extremely low, thus TCP assumes that all losses are due to network congestion which causes router queues to overflow. Therefore, after a loss is detected, the TCP sender reduces its transmission rate to allow router queues to drain, and then slowly increases it so as to gently probe the network [2]. Wireless links are not so reliable though. This causes many UDP applications to fail when used over wireless links with worse quality than expected. TCP applications repeatedly reduce their transmission rate due to frequent wireless errors, attempting to avoid what is (falsely) assumed to be congestion, thus dramatically reducing their throughput. Longer paths suffer more, since end-to-end retransmissions increase recovery delay.

Our work focuses on CC and WLAN systems which are widely available and relatively inexpensive. These systems are differentiated from traditional (geostationary) satellites by their low (terrestrial) propagation delays. Existing CC systems provide wide area coverage at low bit rates, while WLAN systems support higher speeds at low error rates but within smaller areas. WLANs depict losses of up to 1.5% for Ethernet size frames [3], while CC systems suffer from losses of 1–2% for their much shorter (voice optimized) frames [4]. The effects of these losses are dramatic: a 2% packet loss rate over a single WLAN link reduces TCP throughput by half [5].

The performance of real UDP applications (as opposed to UDP based benchmarks) over wireless links has largely been ignored, due to the diversity of UDP applications. Considerable work has been devoted to TCP however, the most popular Internet transport protocol. The goal is to avoid triggering end-to-end congestion recovery due to wireless errors. One way to achieve this is to split TCP connections into one connection over the wireless link and another one over the remainder (presumably, wired part) of the path, bridged by a software agent [6]. Recovery from wireless errors takes place locally over the wireless link. Although this speeds up recovery, it violates end-to-end transport layer semantics. TCP error recovery is also rather inefficient and incapable of taking advantage of link specific optimizations. Splitting TCP connections is also incompatible with IP security which encrypts TCP headers [7].

The alternative to transport layer modifications is local recovery at the link layer. The *Radio Link Protocols* (RLPs) provided by CC systems provide error control customized for the underlying link [4]. Since they only apply a single scheme though, they can be inappropriate for some traffic. For example, retransmissions delay real-time traffic and may interfere with TCP recovery [8]. One way to avoid adverse interactions with TCP is to exploit transport layer information at the link layer. By *snooping* inside TCP headers we can transparently retransmit lost segments when DUPACKs are returned, hiding the DUPACKs from the sender [5]. This scheme avoids link layer error control overhead and outperforms split TCP schemes [5]. However, it only works in the direction from the wired Internet towards the wireless host due to its reliance on TCP DUPACKs. In the reverse direction, DUPACKs are returned too late for local error recovery [9]. It is also incompatible with IP security.

III. Single-Service Link Layer Performance

Link layer schemes are generally preferable to transport layer ones because they provide a local solution to a local problem. They can be customized for the underlying link and deployed transparently to the rest of the Internet. To examine the performance of various link layer error recovery mechanisms in conjunction with a diverse set of applications, over a range of wireless error models, we performed extensive simulations using the Network Simulator version 2 (ns-2) [10]. We present elsewhere the details of the simulation set-up, the experiments and full results [11]. Here we provide a sample of our measurements to motivate the discussion on Quality of Service issues.

We simulated two topologies, depicted in Fig. 1. In the first scenario Wireless Host A and Wireless Host B communicate over two identical but independent wireless links and a LAN link with 10 Mbps speed and 1 ms delay. We also simulated a WAN path, abstracted as a slower, higher delay link [11]. In the second scenario, Base Station A communicates with Wireless Host B over a LAN and a single wireless link. Unlike the symmetric first scenario, in the second one the Base Station is the "server" or "sender," i.e. most data flows from the Base Station to the Wireless Host. However data may also flow in the reverse direction, for example TCP ACKs. All wireless links are WLANs with 2 Mbps speed and 3 ms delay, using 1000 byte packets. Bit errors occur over the WLANs at exponentially distributed intervals with average durations between 2^{14} and 2^{17} bits [5], which lead to packet loss rates of 0.8% to 5.9%. Bit errors influence all packets in both directions of transfer, including TCP ACKs. We also simulated CC links, using a completely different loss model [11].

We tested both TCP and UDP applications so as to examine the benefits of diverse link layer schemes. For TCP, we simulated the *File Transfer Protocol* (FTP), measuring the application level throughput of a 100 MByte transfer. We used the Reno variant of TCP with 500 ms granularity timers. For UDP, we simulated part of a real-time conference, with a single sender (speaker) and a single receiver. We used a speech model where the speaker alternates between talking and silent states with exponential durations averaging 1 s and 1.35 s, respectively [12]. When talking, the speaker sends audio and video at a *Constant Bit Rate* (CBR) of 1 Mbps. We measured the application level packet loss rate and a delay metric consisting of mean packet delay plus twice its standard deviation, so as to account for variable delays, over a 500 s interval.

For TCP, in addition to the *Raw Link* (native) service, we used a full recovery *Selective Repeat* (SR) scheme allowing multiple negative acknowledgments (NACKs) per loss [13], and *Karn's RLP*, a limited recovery scheme which gives up on losses that persist after a number (3 for TCP tests) of retransmissions [4]. We also tested the *Berkeley Snoop* scheme which employs transport layer information for link layer recovery [5]. SR and Karn's RLP treat both TCP data and ACKs in the same manner, while Berkeley Snoop only retransmits TCP data. Fig. 2 shows the throughput achieved over the single wireless link topology by each scheme, for a range of error rates. All figures show metrics averaged from 30 runs plus minimum and maximum values. All enhancement schemes offer significant, but similar, performance gains over Raw Link. While increased loss rates dramatically reduce Raw Link performance, the other schemes exhibit only modest performance drops.

In the corresponding two wireless link topology, the throughput results depicted in Fig. 3 reveal the limitations of TCP aware schemes mentioned above. Berkeley Snoop provides loss recovery only in the direction from the base station to the wireless host. Therefore, wireless losses over one of the two wireless links must be recovered from by TCP. In contrast, both SR and Karn's RLP provide efficient loss recovery regardless

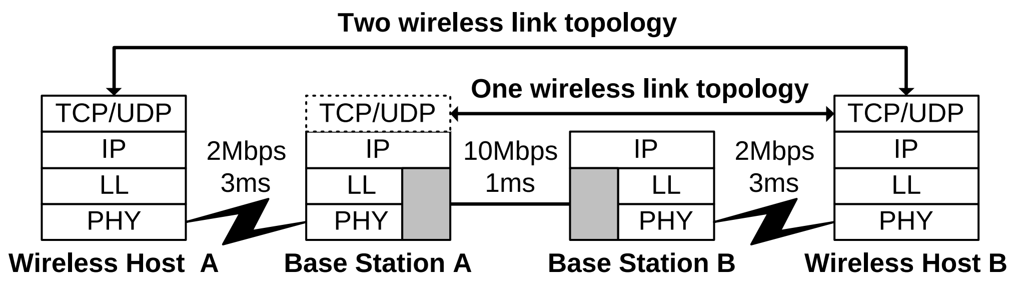

Fig. 1. Simulation topology.

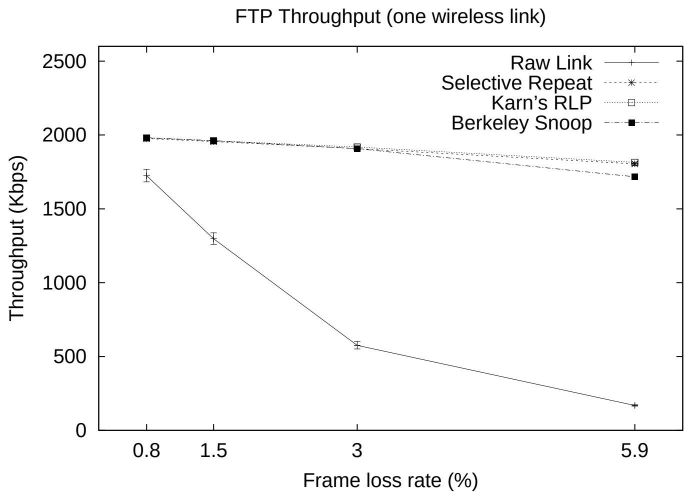

Fig. 2. Stand-alone file transfer over one wireless link: throughput.

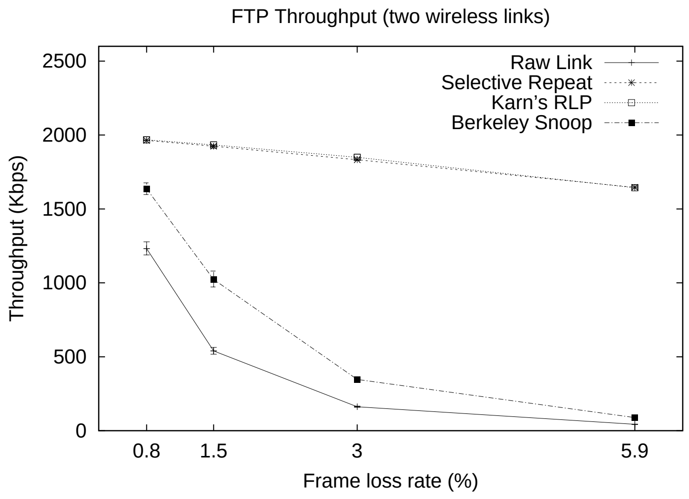

Fig. 3. Stand-alone file transfer over two wireless links: throughput.

of the underlying topology. These schemes perform similarly since the low error rates of WLAN links make persistent errors very rare. As a result, Karn's RLP effectively provides full recovery. Overall, in our FTP tests the TCP unaware link layer schemes (SR and Karn's RLP) improve throughput (compared to Raw Link) by 15–2900%, depending on the topology and native loss rate. Berkeley Snoop depicts topological limitations, which cause it to fail not only over multiple wireless link paths, but also with (bi-directional) interactive applications [11].

For UDP, we tested link layer schemes that offer limited recovery in exchange for reduced delays. In addition to Karn's RLP, we designed an *Out of Sequence* (OOS) RLP which releases packets to higher layers as they arrive. Karn's RLP (and SR) release packets in sequence, thus packets received after a loss have to be buffered until the lost one is retransmitted. We also studied a block based *Forward Error Correction* (FEC) scheme producing one parity packet every 12 data packets [11]. The parity packet is the logical XOR of the data packets, hence one loss can be tolerated per (12 packet) block. Fig. 4 shows the residual loss of each scheme against the native loss rate, for real-time conferencing in the single wireless link topology. Both RLP schemes dramatically reduce losses, even though their retransmission limit is set to 1. The FEC scheme on the other hand depicts gains that do not justify its error recovery overhead.

Fig. 5 shows the delay metrics of each scheme for this scenario. The Raw Link curve is flat since no recovery takes place. While both RLP variants perform similarly at low error rates, as link conditions deteriorate only OOS RLP manages to keep delay low. In sequence delivery causes many packets to be delayed after every loss, thus inflating both average delay and its standard deviation. With multiple wireless links, in sequence delivery can cause packets to be repeatedly delayed even when they are not lost, thus making OOS RLP more attractive. While TCP favors in sequence delivery, OOS RLP is perfectly adequate for real-time applications that use their own resequencing buffers. For the low delay WLAN links, OOS RLP leads to lower delays than the FEC scheme studied, in which recovery has to wait for the parity packet to arrive.

IV. Multi-Service Link Layer Architecture

The results presented briefly above show that link layer error recovery dramatically improves Internet application performance over wireless links. Link layer solutions can be locally deployed and customized for the underlying link, without modifications to the Internet at large. Our results also show however that different schemes work best for the TCP and UDP applications tested. Most likely, other UDP applications with their widely varying requirements will favor different mechanisms. Therefore, what is needed at the link layer is a multi-protocol scheme. To this end, we have developed a *multi-service link layer* architecture, which provides multiple link enhancement services in parallel over a single physical link. Each service fits the needs of a generic class of applications, such as TCP based or real-time UDP based. To simplify service design, the core architecture assigns incoming packets to appropriate services and fairly shares the available bandwidth of the physical link. Therefore, services are isolated and unaware of each other, which allows them to be easily implemented and then added or removed depending on future needs.

We summarize below the design of our architecture, which was originally introduced in [14], while the remainder of the article focuses on its interface with various Internet Quality of Service approaches. Fig. 6 shows an outline of our scheme. Incoming packets are classified and passed to the most appropriate service, based on their sending application. A simple classifier may use the protocol field of the IP header to distinguish between TCP and UDP and the port field of the TCP/UDP headers to determine the application in use. Alternatively, when the *Differentiated Services* (DS) architecture is used for QoS provision at higher layers, the classifier may exploit the DS field of the IP header [15], which remains visible even with IP security [7]. Packets from unrecognized applications are mapped to the default, best-effort, service. Each service operates in isolation, using retransmissions, FEC, or any other mechanism desired, and keeping its private buffers, counters and timers. Outgoing frames are passed to a scheduler which tags each frame with a service number and eventually passes it to the MAC sublayer for transmission. At the receiver, frames are demultiplexed based on their tags and passed to the appropriate service, which may eventually release them to higher layers.

Since each service is allowed to arbitrarily inflate its data stream with error recovery overhead, we must use a frame scheduler to prevent heavily inflated streams from monopolizing the physical link. We chose a *Self Clocked Fair Queueing* (SCFQ) scheduler [16] which can efficiently but strictly enforce the desired bandwidth allocation for each service when the link is loaded. When some services are idle, their bandwidth is shared among the rest in proportion to their original allocations. Fig. 7 gives an outline of the scheduler. The rate table holds the fraction of link bandwidth allocated to each service. We can set these rates statically to reflect policy decisions, or the classifier may dynamically set them to the fraction of incoming traffic that it recently allocated to each service, before error recovery takes place. This guarantees that the best-effort service will receive exactly the same bandwidth as without any link layer enhancements, irrespective of the other services on the link.

Frames awaiting transmission are buffered in service specific queues. A virtual time variable is maintained which is equal to the *time stamp* of the last packet transmitted. To determine the time stamp of an incoming packet, we divide its size by its service rate, and add the result to the time stamp of the preceding frame in its queue. If its queue is empty, we use the current virtual time instead. When the link becomes idle, the frame with the lowest virtual time is dequeued, its time stamp becomes the system's virtual time, and the frame begins transmission. The scheduler organizes all non-empty queues in a heap sorted by the time stamp of their first frame, thus the next frame to transmit is always found at the top. The heap is partially re-sorted when a frame is dequeued for transmission, using the new head of its queue. Empty queues are removed from the heap. They are re-inserted when they receive a new frame, also causing a partial heap re-sort. Thus, each frame requires a simple calculation for its time stamp and $\log_2 n$ operations (for n services) to re-sort the heap when the frame leaves (and, possibly, when

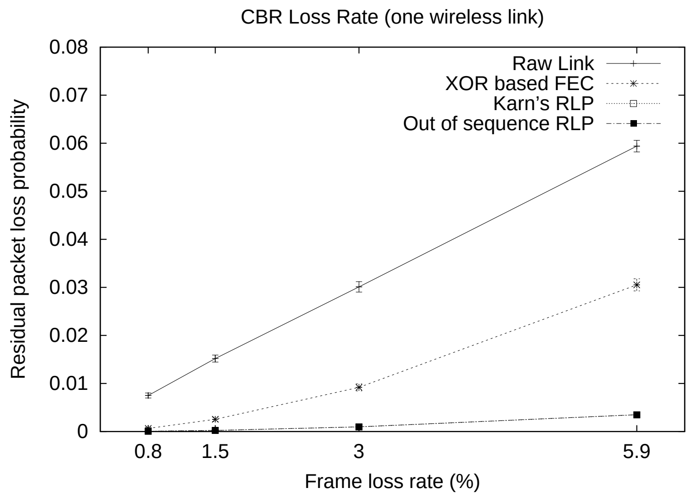

Fig. 4. Stand-alone real-time conferencing over a single wireless link: loss.

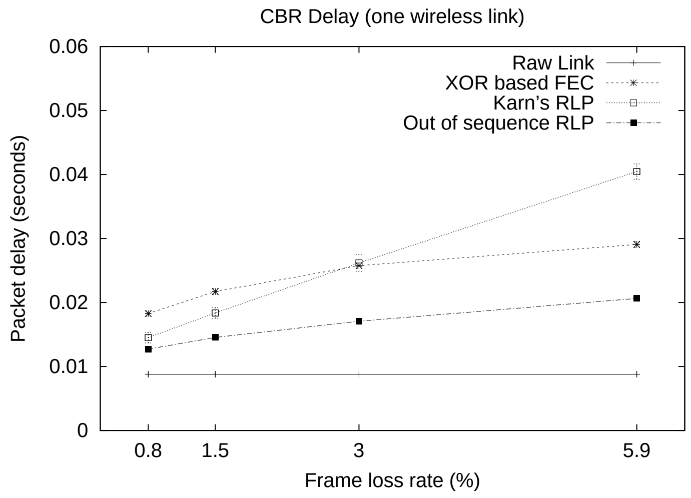

Fig. 5. Stand-alone real-time conferencing over one wireless link: delay.

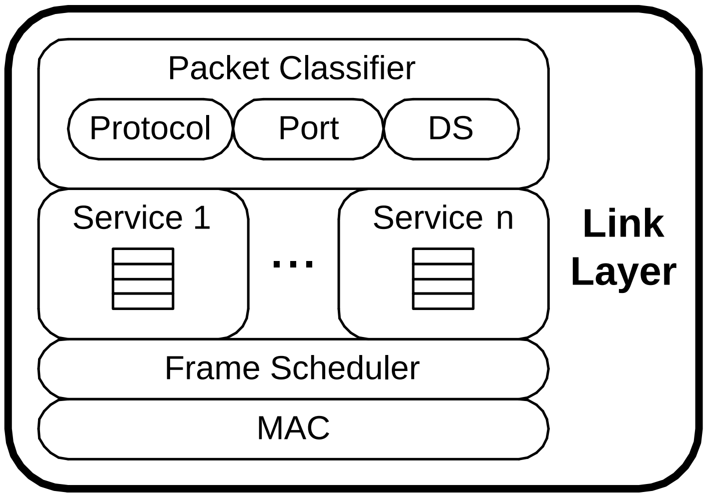

Fig. 6. Multi-service link layer architecture.

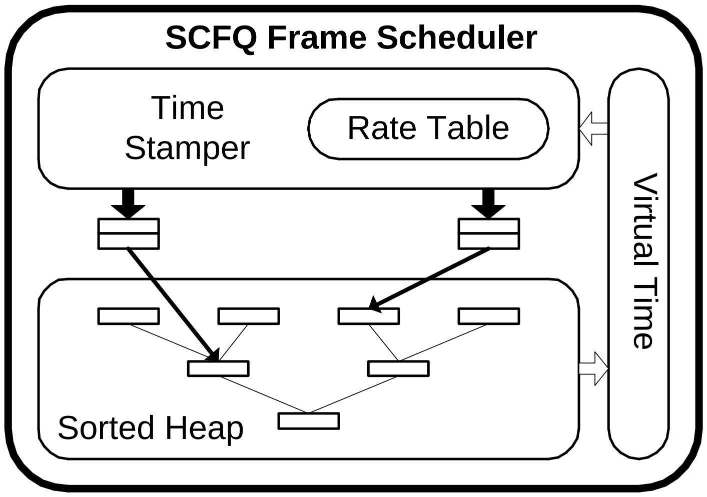

Fig. 7. Self Clocked Fair Queueing frame scheduler.

it enters) the scheduler.

Our multi-service link layer architecture can be incrementally deployed over arbitrary wireless links, transparently to the rest of the Internet. Additional services can be provided by inserting new modules and extending the mappings of the packet classifier, which is reusable over any type of wireless link. Services may be optimized for the underlying link, freely selecting the most appropriate mechanisms for their goals. The scheduler ensures that services will fairly share the link despite their variable overheads, without being aware of each other. Service rates may be set statically or calculated dynamically to match classifier decisions, while applications may be mapped to services either heuristically, or, in the presence of higher layer QoS schemes, intelligently.

V. Multi-Service Link Layer Performance

In order to verify whether our architecture can simultaneously enhance the performance of diverse applications, we repeated the simulations of Sect. III with both applications executing in parallel over the same path but using different link layer schemes. Although the scheduler allocated 1 Mbps to the UDP application, the average bandwidth available to FTP was 1.575 Mbps as real-time conferencing was not constantly active. Fig. 8 shows FTP throughput in the two wireless link topology over various schemes (in parentheses, the scheme used for UDP). The curves are similar to those of Fig. 3, showing that contention with real-time conferencing only results in reduced FTP bandwidth. UDP application bandwidth is effectively protected by the scheduler which limits even the more persistent SR scheme to its fair share of the link.

Residual losses for real-time conferencing were the same as in single application tests, as error recovery is not influenced by contention. The delay metrics, shown in Fig. 9 for the above scenario, were inflated though, an unavoidable effect with our non-preemptive work-conserving scheduler, which managed however to keep delay at reasonable levels. The reduced delays at higher native loss rates with some TCP schemes were due to deteriorating FTP performance which reduced contention. Karn's RLP provided a better balance between FTP throughput and real-time conferencing delay than the more persistent SR scheme. Overall, the multi service link layer approach provided similar gains as in single application tests. FTP performance improved by 11–2425%, depending on the topology and native loss rate, while conferencing delay was kept low as the scheduler prevented TCP from stealing UDP bandwidth.

VI. Best-effort Service Interface

A critical component of a multi-service link layer is the classifier function that maps incoming IP packets to available link services. The lowest standard protocol layer on the Internet, the network layer, provides a single, best-effort, packet delivery service. It is assumed that higher layer protocols can extend this service to satisfy any additional application requirements. Our simulations show however that multiple link layer services are actually needed to enhance Internet application performance over wireless links. Since IP and higher layer protocols are not aware of multi-service links though, they cannot map themselves their requirements to the available link services. To remain compatible with the existing Internet infrastructure, our architecture should perform this mapping transparently, without changing the interface between the link and network layers. Due to the scale and constant evolution of the Internet, such changes take very long to propagate everywhere. Performance should always be at least as good as with a single service, i.e. applications should never be mapped to services that degrade their performance and enhancing the performance of some applications should not degrade the performance of others.

By meeting these requirements, we allow services to be introduced gradually to enhance the performance of some applications, without affecting the rest. Our link layer architecture has single entry and exit points to allow direct integration with existing network protocol stacks. Since we cannot change this interface, the only data we can use for service selection are the contents of the incoming IP packets. In order to match application requirements to available link services we can use a classifier which employs heuristic rules to recognize known applica-

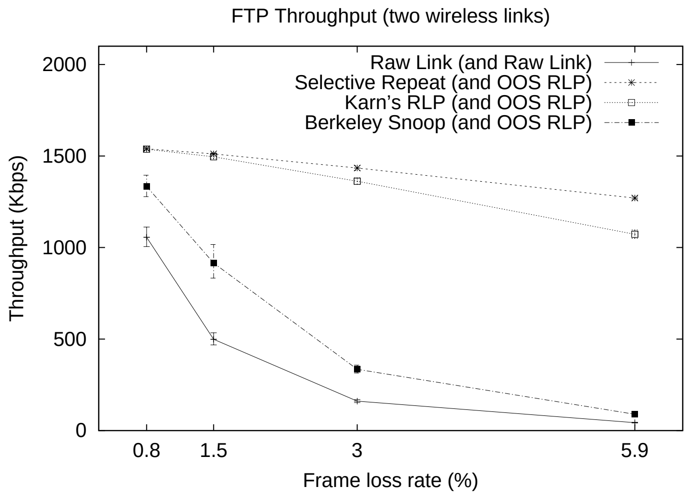

Fig. 8. File transfer and real-time conferencing over two wireless links: throughput.

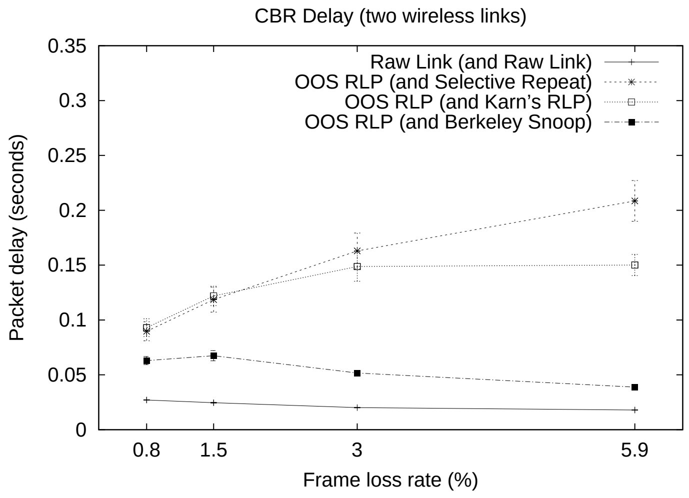

Fig. 9. Real-time conferencing and file transfer over two wireless links: delay.

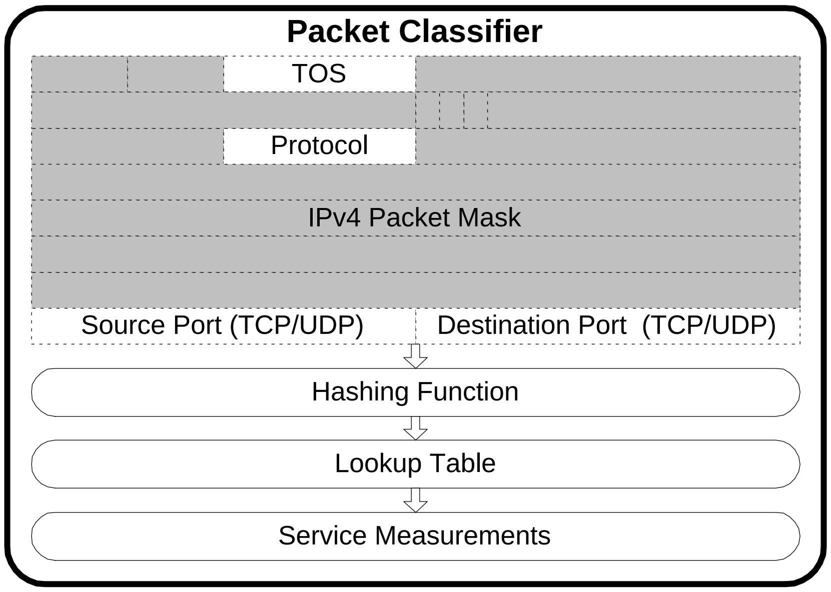

Fig. 10. Heuristic packet classifier.

tions based on IP, TCP and UDP header fields. Fig. 10 shows data flow in this heuristic classifier for IPv4 packets. Headers are masked to isolate the fields used for classification (masked fields are grayed out). These fields pass through a hashing function that produces an index to a lookup table, whose entries point at the available services. Unrecognized applications are mapped to entries pointing at the default (native) link service, which provides the same performance as with a single service link layer.

Whether higher layers perform packet scheduling or not, it is expected that the link layer will allocate bandwidth as if only a single service was offered. For classification to be transparent, the services sharing the link should respect this (implicit) bandwidth allocation, even though each service may introduce arbitrary amounts of error recovery overhead. Thus, after packets are assigned to services, the classifier measures their size to deduce the implicit share of the link allocated to each service. Over regular time intervals, the classifier divides the amount of data assigned to each service i by the total amount of data seen, to get a fraction r_i, where $\Sigma_{i=1}^{n} r_i = 1$, for each of the n services. These fractions are entered into the rate table of the frame scheduler. Unknown applications mapped to the default service are thus allocated at least the same bandwidth as with a single service, while known applications can trade-off throughput, when the link is loaded, for better error recovery. The header mask, hashing function and lookup table are provided by an external administrative module that is aware of application requirements, header fields and link services.

Heuristic packet classification starts with the protocol field of the IP header, which indicates TCP, UDP, or another protocol. All TCP applications can be mapped to a single TCP oriented link service, implemented by one of the schemes mentioned above. UDP applications have more diverse requirements though, hence decisions must be made on a per application basis. Known UDP applications can be recognized by looking at the source and destination port fields of the UDP header. Many applications use *well-known ports* to communicate, for example to allow peer applications to be located at remote hosts. Thus, the protocol and port fields can be used to recognize applications, so that their data may be mapped to the most appropriate service. Another field that may be used is the *Type of Service* (TOS) field of the IPv4 header. Originally, four bits were defined to indicate preference for reduced delay, reduced cost, increased throughput and increased reliability, while three more bits were defined to indicate packet priorities. The TOS field, however, is rarely used and it is being redefined to support Differentiated Services. This also applies to the *Traffic Class* field of the IPv6 header.

A significant drawback of heuristic classifiers is the considerable effort required to construct them. Applications must be manually matched to the services available over each type of wireless link whenever new applications or services are added. Many applications will not be recognized, not only due to the large number of existing and future applications, but also because many applications do not use well-known ports. Although some higher layer QoS schemes combine the transport protocol, source/destination port and host address fields to classify packets [17], setting up and maintaining state for all these combinations requires end-to-end signaling and considerable storage, both of which are not available at the link layer. Another important problem with heuristic classifiers is that IP security mechanisms encrypt the source/destination ports of TCP and UDP headers and replace the value of the protocol field with the identifier of the IP security protocol [7]. This leaves only the TOS field visible, which is currently inadequate to describe application requirements.

VII. Differentiated Services Interface

The single best-effort service provided by IP is becoming inadequate as real-time applications migrate from the circuit-switched telephone network with its explicit delay guarantees to the packet-switched Internet. For these applications to exploit the reduced costs promised by statistical multiplexing, some type of performance guarantees must be introduced into the Internet. Users would presumably pay more for better Internet service, if it was still cheaper to use the Internet rather than a circuit-switched network for the same task. The main issue for Internet QoS is generally considered to be congestion control. Applications may not be able to get the throughput they need due to contention for link resources and their end-to-end delay may increase due to queueing delays at congested routers. When router queues overflow, data loss also occurs. UDP applications have to deal with these losses themselves, while TCP applications face additional delays during end-to-end recovery.

One scheme for Internet QoS provision is the *Integrated Services* architecture [18], which has been criticized in two ways. First, it must be widely deployed over the Internet to be useful, since its guarantees rely on actions at every router on a path. Second, it mandates resource reservations on a per flow basis, where a flow is defined as a data stream between two user processes with fixed QoS requirements. Since a huge number of flows exists, the global scalability of any QoS scheme based on per flow state is doubtful. In IPv6 a 20 bit flow label is defined in the IP header to identify flows between two hosts, while host addresses are expanded to 128 bits. The only solution to this scalability problem is to aggregate flow state, but it is unclear how to do so.

An alternative scheme that attempts to avoid these limitations is the *Differentiated Services* architecture [15], in which flows are aggregated into a few classes, either when entering the network, or when crossing network domains. At these points only, flows may be rate limited, shaped or marked to conform to specific traffic profiles. These are negotiated between users and network providers or between neighboring domains. Within a domain, routers only need to select a *Per-Hop Behavior* (PHB) for each packet, based on its class, denoted by the 8-bit Differentiated Services (DS) field of the IP header. This subsumes both the IPv4 TOS and the IPv6 Traffic Class fields. State aggregation into a few classes means that this scheme scales well, but the guarantees that may be provided are not as fine grained as with Integrated Services.

This architecture intentionally leaves the definition of PHBs open, to allow experimentation with different approaches. For example, the *expedited forwarding* PHB provides a minimum amount of bandwidth at each router, for traffic that was rate limited when entering the network or the domain so as not to exceed this bandwidth. This PHB provides low delay and loss by eliminating congestion for the class, offering a virtual leased line service. As another example, the *assured forwarding* PHB group defines a number of service classes, with each one allocated a specific share of the bandwidth. Within each class, packets may have multiple levels of drop preference. Besides scheduling so as to satisfy the bandwidth requirements of each class, when a class is congested routers should first drop the packets with highest drop preference. Flows are marked with higher drop preference levels when they exceed the traffic profile of their class, rather than being rate limited. Both PHBs may be implemented by many scheduling mechanisms and queue management schemes.

The services provided by this architecture depend on the available PHBs and are meant to provide generic QoS levels, not application specific guarantees, hence the mapping of traffic classes and not flows to PHBs. Only network entry points are aware of both application requirements and PHB semantics so as to perform flow aggregation. Similarly, only domain entry points are aware of the semantics of PHBs available in their neighboring domains so as to perform appropriate translations. Traffic policing, shaping and marking, is also only performed at those points. For neighboring domains, traffic profiles should be relatively static as they represent large traffic aggregates, while at network entry points they could be frequently modified by the user. This is far more economical than the flow specific signaling of Integrated Services.

Differentiated Services and multi-service link layers solve orthogonal but complementary problems. Differentiated Services are concerned with congestion and its impact on throughput, delay and loss. The services offered are based on link independent PHBs, provided by IP level packet scheduling and queue management mechanisms. Multi-service link layers are concerned with recovery from link errors, customized to each type of application. The error recovery mechanisms used are link dependent and local, thus they cannot be standardized into link independent PHBs. The frame scheduling provided only protects services from each other by mirroring higher layer allocations, it does not offer end-to-end guarantees. By only providing Differentiated Services over wireless links we can offer to applications a nominal IP level QoS, but their actual performance will be limited by link losses. Even if we reserve wireless link bandwidth for a TCP traffic class for example, only a small fraction of it will be used due to losses and inefficient TCP (end-to-end) error recovery. Multi-service link layers provide adequate recovery to fully utilize wireless links, but they also need higher layer guidance to allocate link bandwidth.

These two architectures are excellent complements to each other. Differentiated Services provide congestion control, using packet scheduling and queue management, while multi-service link layers add application dependent error control, respecting higher layer scheduling despite the introduction of recovery overhead. They both offer a few services at each node (PHBs or link mechanisms) for aggregated traffic classes with common requirements. They can be combined by extending the DS field to also specify the error requirements of each traffic class. For example, a traffic class with a reserved amount of bandwidth could be subdivided into two subclasses with different error recovery requirements by using one more bit of the DS field. Each subclass would be mapped to an appropriate link layer service. DS bits are set when flows are aggregated into classes. Applications could indicate their requirements when injecting their traffic into the network, with boundary routers translating them to local equivalents when required. The result is a simplified multi-service classifier, shown in Fig. 11 for IPv6 packets. The DS field is isolated via a header mask, and then a hashing function plus a lookup table map it to an appropriate service. The same procedure can be used with IPv4 headers, where the DS field is the original TOS field instead of the IPv6 Traffic Class field.

Such a classifier does not rely on multiple header fields and complex rules to determine application requirements. A single field is used with well-defined semantics. New applications may be mapped to existing classes if their requirements are similar to those of previously mapped applications. More importantly, the DS field is not obscured by IP security mechanisms, it is visible even in encrypted packets. Since the Differentiated Services module performs scheduling and queue management, traffic entering the multi-service link layer already obeys the required bandwidth allocations. For example, two TCP traffic subclasses belonging to separate DS classes may be rate limited in different ways at the IP level. At the link layer, however, they are already shaped as needed, hence they can share the same service and frame scheduler queue without introducing congestion. Thus, regardless of the number of DS traffic classes used, the multi-service link layer must only maintain a single instance of each service. The service rates can be set in two ways. If the subclasses of each DS traffic class have separate bandwidth allocations at the IP level, then the bandwidth for all subclasses mapped to the same link service is added to set its service rate. If subclasses share a common bandwidth pool within each class, then a service measurements module can be inserted after the lookup table, as in Fig. 10, to automatically determine service rates.

VIII. Advanced Quality of Service Interface

The performance of the various services available at each wireless link can vary widely. Different links may favor different error recovery schemes, with the performance of each scheme varying over time due to unpredictable environmental conditions. If a characterization of the end-to-end performance of a network path is provided, applications can verify that a given end-to-end service is suitable for their needs [19]. In addition, when path characteristics change significantly, updated characterizations enable adaptive higher layers to modify their policies. Hierarchical cellular systems for example are composed of multiple overlaid cellular systems: satellites cover the whole globe, cellular systems cover populated areas, and indoor WLANs cover buildings. In these systems, in addition to *horizontal* handoffs between adjacent cells of the same system, the user can perform *vertical* handoffs between different systems. Horizontal handoffs may cause changes to error behavior and traffic load, while vertical handoffs may alter all wireless link characteristics. To describe the services offered over network paths that are so diverse, we must dynamically discover what

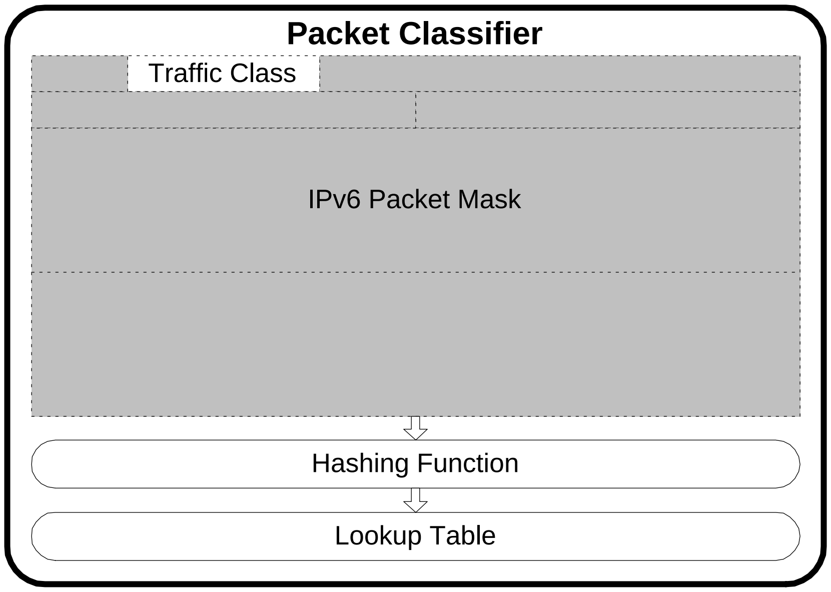

Fig. 11. Differentiated Services packet classifier.

is provided at each link. This can be achieved if each service dynamically characterizes its performance with a set of standardized metrics. Dynamic characterization means that performance may be evaluated as often as needed, while metric standardization means that higher layers will be able to assess the performance of arbitrary services without any knowledge of the link layer mechanisms employed. This would enable end-to-end QoS modules to compose local link metrics into end-to-end path metrics.

We have defined three link independent metrics, reflecting the possible trade-offs available to each error recovery scheme: goodput, loss and delay. Metrics are calculated dynamically over an implementation dependent interval. Reported metrics may be smoothed by using a weighted average of the latest calculated value and the previous reported value, similar to TCP round trip delay estimates. *Goodput* (g_i, for service i) is the ratio of higher layer data transmitted during the measurement interval to link layer data transmitted, including all overhead. The amount of higher layer data transmitted may differ from the amount received due to residual losses. Goodput can be calculated at the sender without any receiver feedback. *Loss* (l_i) is the ratio of higher layer data lost to higher layer data transmitted (lost plus received). Loss is calculated by the receiver based on the sequence of data released to higher layers. It depicts the residual loss rate after link layer error recovery. It can be greater than zero for limited recovery schemes. *Delay* (d_i) is the one way average delay (in seconds) for higher layer packets. Delay can be estimated at the receiver based on knowledge of the implemented recovery scheme and wireless link characteristics. Retransmission schemes could add one round trip delay estimate for each retransmission to their one way delay estimate. A FEC scheme could instead add the interval between the loss of a recovered frame and its reconstruction from parity data to its one way delay estimate.

The delay metric only denotes the error recovery delay of a service, since there is no congestion inside the link layer. It can be added to IP level queueing delay to give the total delay for each node, which can be used to estimate end-to-end delays incorporating both congestion control and wireless error recovery. Delay sensitive traffic subclasses should choose the lowest delay service whose residual loss falls within their tolerance limits. Goodput can be combined with loss to get *Effective Goodput* (e_i), defined as $e_i = g_i * (1 - l_i)$, or, the ratio of higher layer data received to link layer data transmitted. Essentially, e_i shows how much of the bandwidth allocated to a service is used by the data actually received, after subtracting error recovery overhead and residual losses. If the link bandwidth is B and service i is allocated a service rate r_i in the scheduler, then its throughput is $B * r_i * e_i$. This may be used to estimate the throughput for each service given a set of service rates, or to calculate the service rate needed to achieve a target throughput for a particular service. The reason for using goodput instead of throughput for service characterization is exactly that goodput, unlike throughput, can be used to predict service behavior with different rate allocations. All these metrics are summarized in Table I.

These metrics may be used at multiple layers to serve different needs, as shown in Fig. 12. The physical layer provides hardware dependent information, such as the fixed one way delay, that may be used by link layer services to provide their link independent g_i, l_i and d_i metrics. At the network layer, scheduling mechanisms such as *Class Based Queueing* (CBQ) may use those metrics to set the bandwidth allocations for each service. End-to-end QoS schemes may use the *Resource ReSerVation Protocol* (RSVP) to gather information about node services so as to estimate end-to-end path characteristics. These can in turn be used by both transport protocols and applications to adapt their operation to prevailing conditions. For example, TCP may limit its congestion window to respect available bandwidth limitations, or, video conferencing applications may select encod-

TABLE I
SERVICE CHARACTERIZATION METRICS.

Name	**Symbol**	**Definition**
Goodput	g_i	$\frac{\text{higher layer data transmitted}}{\text{link layer data transmitted}}$
Loss	l_i	$\frac{\text{higher layer data lost}}{\text{higher layer data transmitted}}$
Delay	d_i	Average packet delivery delay
Effective Goodput	$e_i = g_i * (1 - l_i)$	$\frac{\text{higher layer data received}}{\text{link layer data transmitted}}$

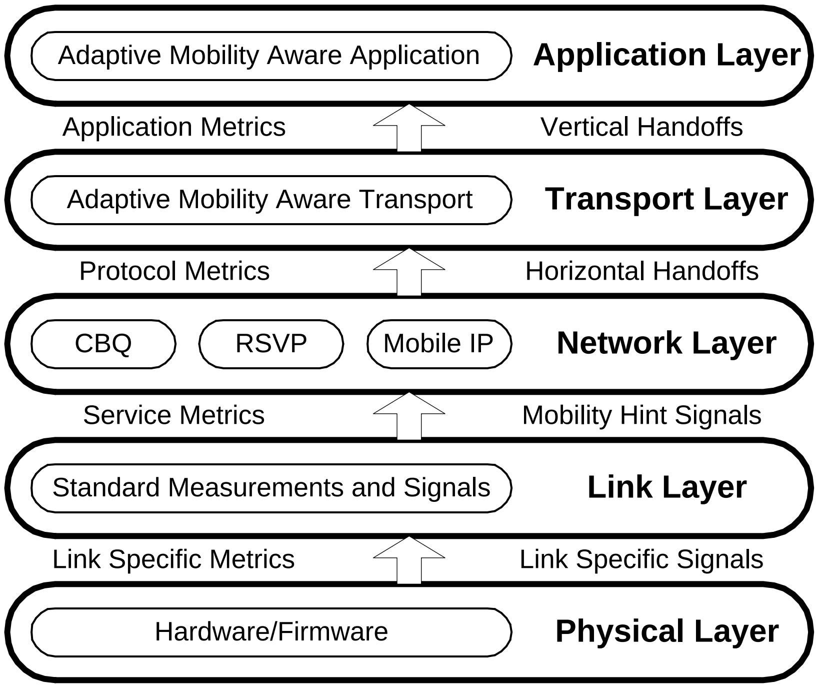

Fig. 12. Propagation of service measurement and mobility feedback.

ing schemes that can deal with the residual loss rate of the path. Metrics can be refined at each layer, as in the network layer where local link metrics are used to compose end-to-end path metrics.

To assist higher layers in dealing with mobility, we can extend this interface to provide *mobility hints* to interested parties via upcalls, as shown in Fig. 12. The link layer can combine hardware signals with its own state to detect events like connections and disconnections. If a handoff is taking place, higher layers will receive one disconnection and one connection upcall from different links. These link independent upcalls can be used by the IP mobility extensions to allow fast detection of handoffs, instead of relying on periodic network layer probes [20]. Higher layers may be notified by the network layer of horizontal and vertical handoffs via further upcalls. TCP may be notified of pending horizontal handoffs to temporarily freeze its timers and avoid timeouts during disconnection intervals. A video conferencing application may be notified of vertical handoffs so as to change the encoding scheme used to a higher or lower resolution one, depending on available bandwidth. The characteristics of the new path can be discovered by using end-to-end QoS provisioning mechanisms to query the metrics exported by the new wireless link of the path.

Our QoS interface was designed to fit the needs of *One Pass With Advertising* (OPWA) [19] resource reservation mechanisms. One pass schemes cannot specify a desired service in advance as they do not know what is available on a path [17], thus the resources reserved may provide inadequate service. Two pass schemes specify a service in advance but make very restrictive reservations on the first pass, relaxing them on a second pass [21]. Such reservations may fail due to tight restrictions on the first pass. In an OPWA scheme, an advertising pass is first made to discover the services available on the path, and then a reservation pass actually reserves the resources needed for the selected service. Our interface allows OPWA schemes to discover the throughput, delay and loss restrictions imposed by wireless links, by looking at local service metrics. After that information is gathered, applications may choose a service, and the reservation pass can set up appropriate state so as to provide it. Mobility hints notify higher layers that they should revise their path characterizations after handoffs, thus fitting the needs of adaptive mobility aware applications which can adapt their operation based on resource availability.

A related proposal has been made in the context of IP mobility: a router could notify hosts communicating with a wireless host of new link characteristics after a handoff [22]. This approach however does not support link characterization between handoffs or multiple services. Another related proposal specifies an adaptation interface for mobility aware applications that notifies applications when available bandwidth deviates from a given range [23]. An application supporting multiple encodings of its data would select a bandwidth range for each, switching encodings whenever the available bandwidth moved into another range. This interface is not appropriate for the link layer, as it requires end-to-end signaling and per application state, but it can be easily implemented using our mobility hints. A QoS management module at each host would accept bandwidth range requests from local applications. Instead of monitoring the link, it would only check its data base after receiving a mobility hint, in turn notifying the applications whose ranges had changed.

IX. Conclusions

While extending the Internet over wireless links is straightforward, application performance over wireless links using the traditional Internet protocols has been disappointing due to channel impairments and adverse interactions between protocol layers. Different applications favor different error control approaches, thus we developed a link layer architecture that provides multiple services simultaneously over a single link, allowing the most appropriate link service to be used by each traffic class. Our approach is easy to optimize for each underlying wireless link and efficient to operate. It can be locally deployed, transparently to the rest of the Internet, and it is easy to extend to address future requirements. It can be incorporated into the Internet QoS architecture in three stages. First, in the current best-effort only Internet, we can employ heuristic classifiers to select services provided over individual multi-service links to enhance the performance of recognized applications. Second, in the context of the Differentiated Services architecture, which focuses on end-to-end congestion control, multi-service link layers can provide application dependent error control, respecting higher layer scheduling decisions despite the introduction of error recovery overhead. Finally, multi-service link layers can support an advanced QoS application interface, offering dynamic end-to-end service discovery and synthesis.

Acknowledgments

The authors gratefully acknowledge the support of the General Secretariat of Research and Technology of Greece under grant 99ED193. A preliminary version of this article was presented at the International Workshop on QoS in Multiservice IP Networks, Rome, Italy, January 2001 [1].

References

[1] G. Xylomenos, G. C. Polyzos, Quality of Service issues in multi-service wireless Internet links, in: Proc. International Workshop QoS-IP 2001, Lecture Notes in Computer Science, vol. 1989, M. Ajmone Marsan, A. Bianco, (Eds.), Springer, Berlin, 2001, pp. 347–365.

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

[3] G. T. Nguyen, R. H. Katz, B. Noble, M. Satyanarayanan, A trace-based approach for modeling wireless channel behavior, in: Proc. of the Winter Simulation Conference, 1996, pp. 597–604.

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

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, A comparison of mechanisms for improving TCP performance over wireless links, in: Proc. of the ACM SIGCOMM ’96, 1996, pp. 256–267.

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

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

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

[9] G. Xylomenos, G. C. Polyzos, Internet protocol performance over networks with wireless links, IEEE Network 13 (1999) 55–63.

[10] UCB/LBNL/VINT Network Simulator - ns (version 2), available at http://www-mash.cs.berkeley.edu/ns.

[11] G. Xylomenos, Multi Service Link Layers: An approach for enhancing Internet performance over wireless links, Ph.D. thesis, University of California, San Diego, available at http://www-cse.ucsd.edu/groups/csl/pubs/phd.html (1999).

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

[13] P. T. Brady, Evaluation of multireject, selective reject, and other protocol enhancements, IEEE Transactions on Communications 35 (1987) 659–666.

[14] G. Xylomenos, G. Polyzos, Link Layer Support for Quality of Service on Wireless Internet Links, IEEE Personal Communications 6 (1999) 52–60.

[15] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for Differentiated Services, RFC 2475 (December 1998).

[16] S. Golestani, A self-clocked fair queueing scheme for broadband applications, in: Proc. of the IEEE INFOCOM ’94, 1994, pp. 636–646.

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, RSVP: A new resource reservation protocol, IEEE Network 7 (1993) 8–18.

[18] D. D. Clark, S. Shenker, L. Zhang, Supporting real-time applications in an integrated services packet network: architecture and mechanism, in: Proceedings of the ACM SIGCOMM ’96, 1996, pp. 243–254.

[19] S. Shenker, L. Breslau, Two issues in reservation establishment, in: Proceedings of the ACM SIGCOMM ’95, 1995, pp. 14–26.

[20] C. Perkins, IP mobility support, RFC 2002 (October 1996).

[21] D. Ferrari, D. C. Verma, A scheme for real-time channel establishment in wide-area networks, IEEE Journal on Selected Areas in Communications 8 (1990) 368–379.

[22] D. B. Johnson, D. A. Maltz, Protocols for adaptive wireless and mobile networking, IEEE Personal Communications 3 (1996) 34–42.

[23] B. D. Noble, M. Price, M. Satyanarayanan, A programming interface for application-aware adaptation in mobile computing, in: Proceedings of the 2nd USENIX Symposium on Mobile and Location-Independent Computing, 1995, pp. 57–66.