Quality of Service Issues in Multi-Service Wireless Internet Links

George Xylomenos and George C. Polyzos
{xgeorge,polyzos}@aueb.gr
Department of Informatics
Athens University of Economics and Business
Athens, Greece 10434

***Abstract*— Internet application performance over wireless links is disappointing, due to wireless impairments and their adverse interactions with higher protocol layers. In order to effectively address these problems without the need for global protocol upgrades, we focus on link layer enhancement schemes. Simulations reveal that different schemes work best for different applications. We have thus developed a multi-service link layer architecture that can simultaneously enhance the performance of diverse applications by supporting multiple link mechanisms concurrently. Simulations confirm that this architecture provides dramatic performance improvements. The architecture can be embedded in various ways into the Internet. A critical issue for Quality of Service support over wireless links is the unpredictability of available resources. Our approach is based on fair sharing of the link before any measures are taken to improve the performance of individual traffic classes. This approach turns over the error control tradeoff to the applications themselves.**

I. Introduction

The Internet is always quick to adopt new communications technologies, largely due to the physical layer independent design 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 of the wireless communications sector. *Cellular Telephony* (CT) is spreading worldwide and evolving towards 3G systems, while *Wireless Local Area Networks* (WLANs) are becoming inexpensive and conformant to international standards. Due to both physical and economic limitations however, wireless links consistently lag behind wired ones in performance.

The popularity of these wireless systems has generated considerable interest in their integration with the existing Internet. Even though satellites have long been a part of the Internet, most higher layer protocols and applications make assumptions about link performance that cannot be met by terrestrial CT and WLAN links. Thus, although providing IP services over wireless links is easy, their performance is disappointing. Since the Internet is evolving towards supporting *Quality of Service* (QoS) in order to enable new applications such as real-time multimedia communications, improving wireless link performance becomes even more critical.

This article presents an architecture that improves the performance of diverse Internet applications and extends QoS provision over wireless links. In Sect. II we outline the problem and review previous approaches, arguing for a link layer solution. Our simulations show that different applications favor different enhancement schemes. In Sect. III we present a *multi-service link layer* architecture that simultaneously enhances the performance of diverse applications by supporting multiple link mechanisms in parallel, without any changes to the rest of the Internet. The remainder of the article discusses how our architecture fits into the context of Internet QoS approaches: Sect. IV covers the existing best-effort service, Sect. V the Differentiated Services architecture, and Sect. VI a dynamic service discovery architecture.

II. Background

IP provides unreliable packet delivery between any two network hosts: packets may be lost, reordered or duplicated. Applications can use the *User Datagram Protocol* (UDP) for direct access to this service. Some UDP applications assume 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 introduce redundancy in their data to tolerate some errors without the need for slow end-to-end retransmissions.

The majority of applications however require complete reliability. Those usually employ the *Transmission Control Protocol* (TCP), which provides a reliable byte stream service. TCP breaks the application data into segments which are reassembled by the receiver. The receiver generates cumulative acknowledgments (ACKs) for those segments received in sequence, with duplicate acknowledgments (DUPACKs) for reordered ones. Although a lost data segment leads to a DUPACK when the next segment arrives, since packets may be reordered by IP, only after multiple (usually 3) DUPACKs are returned does the sender retransmit the (apparently lost) next segment in sequence. The sender also tracks the round trip delay of the connection, so that if a segment is not acknowledged on time it is retransmitted on an end-to-end basis [1].

Since data corruption is extremely rare in wired links, TCP assumes that all losses are due to congestion. Thus, after a loss the sender reduces its transmission rate to allow router queues to drain, and then gradually increases it so as to gently probe the network [1]. Wireless links are considerably less reliable than wired ones though. This causes UDP applications to fail when

used over wireless links with worse quality than expected. TCP applications on the other hand repeatedly reduce their transmission rate when faced with frequent wireless errors, so as to avoid what is (falsely) assumed to be congestion, thus dramatically reducing their throughput. Longer paths suffer more, since end-to-end retransmissions increase delay, a critical issue for interactive applications.

We have focused on CT and WLAN systems which are widely available and relatively inexpensive. The low (terrestrial) propagation delays of such systems differentiate them from traditional (geostationary) satellites. Present CT systems support low bit rates in the wide area while WLAN systems support higher speeds and lower error rates. *Personal Communications Systems* (PCS), the evolution of CT, will provide higher bit rates using smaller coverage cells. WLANs suffer from losses of up to 1.5% for Ethernet size packets [2], while CT systems suffer from losses of 1-2% for their much shorter (voice oriented) frames [3]. The effects of such losses are dramatic: a 2% packet loss rate over a single WLAN link reduces TCP throughput by half [4].

The performance of UDP applications over wireless links has largely been ignored, mainly due to the diversity of UDP applications. Considerable work has been devoted to TCP however, as it is 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 (wired part) of the path, bridged by an agent [5]. Recovery from wireless errors is performed locally over the wireless link. Although this speeds up recovery, it violates transport layer semantics. In addition, TCP error recovery is inefficient and unable to take advantage of link specific optimizations. Split TCP connections are also incompatible with IP security which encrypts TCP headers [6].

The main alternative to transport layer solutions is local error recovery at the link layer. The Radio Link Protocols (RLPs) provided by CT systems offer error control customized for the underlying link [3]. Since they always apply the same scheme though, they may not be appropriate for some traffic. For example, retransmissions may delay real-time traffic or interfere with TCP recovery [7]. One way to avoid such adverse interactions is to exploit transport layer information at the link layer. By *snooping* inside TCP segments we can transparently retransmit lost segments when DUPACKs are returned, hiding the DUPACKs from the sender [4]. This scheme avoids link layer error control overhead and outperforms split TCP schemes [4]. However, it works only in the direction from the wired Internet towards the wireless host due to its reliance on TCPDUPACKs (in the reverse direction, DUPACKs are returned too late for local error recovery) [8] and it is also incompatible with IP security.

Despite their limitations, link layer schemes are preferable to higher layer ones because they provide a local solution to a local problem. Therefore, they can be customized for the underlying link and deployed transparently to the rest of the Internet. In order to examine the performance of various link layer enhancement schemes 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) [9]. We thus studied a wide range of parameters under controlled conditions. The details of the simulation set-up, the experiments and the results are presented elsewhere [10]. Here we only provide a small sample of our results to motivate the discussion on Quality of Service issues.

We simulated two topologies, both depicted in Fig. 1. In the first scenario two Wireless Hosts communicate over two identical but independent wireless links and a single LAN link with 10 Mbps speed and 1 ms delay.[1] In the simpler second scenario, a Wireless Host communicates with a Base Station over a single LAN and a single wireless link. Unlike the former scenario which is symmetric, in the latter one the Base Station is the "server" or "sender," i.e. most data flows from the Base Station towards the Wireless Host. However data may also flow in the reverse direction, for example TCP ACKs. The wireless links are WLANs with 2 Mbps speed and 3 ms delay, using 1000 byte packets. They suffer from bit errors which occur at exponentially distributed intervals with average durations between 2^{14} and 2^{17} bits [4], which lead to packet loss rates of 0.8% to 5.9%.[2]

We tested both TCP and UDP applications so as to examine the benefits of various 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 TCP Reno implementation of ns-2 with 500 ms granularity timers. For UDP, we simulated real-time conferencing 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 [11]. 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, to account for the effects of variable delays, over a 500 s interval.

For TCP, in addition to the *Raw Link* (native) service, we studied a full recovery *Selective Repeat* (SR) scheme which allows multiple negative acknowledgments (NACKs) per loss [12], and *Karn's RLP*, a limited recovery scheme which gives up on losses that persist for a number (3 for TCP tests) of retransmissions [3]. We also tested the *Berkeley Snoop* scheme which employs transport layer information for link layer recovery [4]. Our FTP tests show that 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 however fails in the two wireless link scenarios, as it is unable to retransmit from the Wireless Host towards the Base Station.[3]

For UDP, we tested schemes that provided limited recovery in exchange for lower delays. In addition to Karn's RLP, we designed an *Out of Sequence* (OOS) RLP, which releases packets to higher layers as they arrive, and not in sequence (as in SR and Karn's RLP). We also studied a block based *Forward Error*

[1] We also simulated a WAN path, which is abstracted as a slower, higher delay link.

[2] We also simulated slower CT and PCS links, with completely different loss models.

[3] This limitation is even more apparent with (bi-directional) interactive applications.

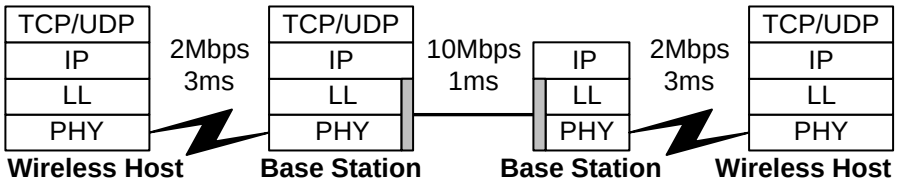

Fig. 1. Simulation topology

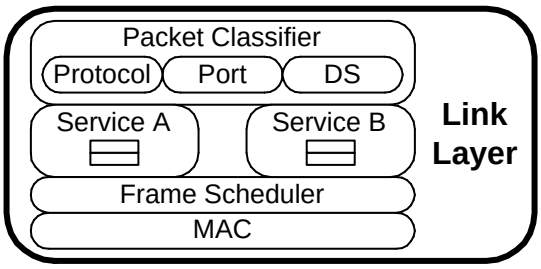

Fig. 2. Multi-service link layer architecture

Correction (FEC) scheme with 1 parity for every 12 data packets [10]. Our real-time conferencing tests show that both RLP schemes dramatically reduce losses, even with 1 retransmission per loss, unlike the FEC scheme whose gains do not justify its overhead. The OOS RLP variant significantly reduces the delay of Karn's RLP, with gains that become more attractive with multiple wireless links. While TCP favors in sequence delivery, OOS RLP is perfectly adequate for real-time applications with their own resequencing buffers. For the low delay WLAN links, OOS RLP actually leads to lower delays than the FEC scheme studied.

III. Multi-Service Link Layer Architecture

The results presented briefly above (see [10] for additional details) 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 any changes to the rest of the Internet. Our results also show however that different schemes work best for the TCP and UDP applications tested, and it is very likely that other applications will favor quite different schemes. Therefore, what we need at the link layer is a flexible multi-protocol approach. 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 requirements of a generic class of applications, such as TCP based or real-time UDP based. To simplify service design, the architecture assigns incoming packets to services and fairly shares the available bandwidth. As a result, services are isolated and unaware of each other, which allows them to be easily added and removed as new needs arise.

We summarize below the design of our architecture, while the following sections focus on its interface with various Internet Quality of Service approaches.[4] Figure 2 gives an outline of our scheme. Incoming packets are classified and passed to the most appropriate service, based on their application. A simple classifier may use the IP protocol field to distinguish between TCP and UDP and the TCP/UDP port field 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 IP DS field [14], which is visible even with IP security. Packets that cannot be matched to any enhanced service are mapped to the default, best-effort, service. Each service operates in isolation, using retransmissions, FEC, or any other mechanism desired, and keeping its own buffers and timers, that may be optimized for the underlying link. 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 passed to the appropriate service (based on their tags), which may eventually release them to higher layers.

Since each service is free to arbitrarily inflate its data with error recovery overhead, we must use a frame scheduler to prevent heavily inflated data streams from monopolizing the physical link. We chose a *Self Clocked Fair Queueing* (SCFQ) scheduler [15] which can strictly enforce the desired bandwidth allocation for each service when the link is loaded. When some services are idle, their bandwidth is proportionately shared among the rest. Figure 3 gives an outline of the scheduler. The rate table shows the fraction of the link allocated to each service. We can set these rates statically, or the classifier may dynamically set them to equal the fraction of incoming traffic that it allocates to each service, before error recovery takes place. In this case, the best-effort service will receive exactly the same bandwidth as without any link layer enhancements.

Frames awaiting transmission are buffered in service specific queues. The scheduler maintains a virtual time variable 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 it to the time stamp of the previous frame in its queue. If its queue is empty, we use the current virtual time instead. When the link is freed, the frame with the lowest virtual time is dequeued, the system virtual time is updated, and the frame starts transmission. The scheduler organizes all service queues in a heap sorted by the time stamp of their first frame, so that the next frame to transmit is always at the top. The heap is re-sorted when a frame is dequeued for transmission or when an empty queue receives a new frame. 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 it enters) the scheduler.

[4] The multi-service link layer architecture was briefly introduced in [13].

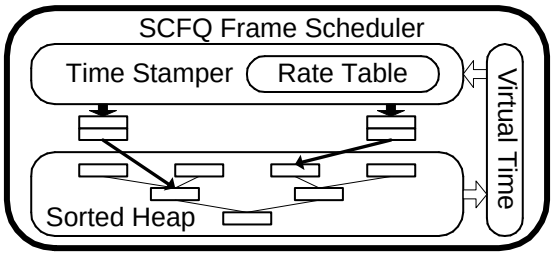

Fig. 3. Self Clocked Fair Queueing frame scheduler

Our multi-service link layer architecture can be locally deployed, 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 may be reused over any wireless link. Services may be optimized for the underlying link, freely selecting the most appropriate mechanisms for their goals. The scheduler ensures that services fairly share the link despite their variable overheads. Service rates may be set statically or dynamically, while applications may be mapped to services either heuristically, or, in the presence of higher layer QoS schemes, intelligently. To verify our claims that the multi-service link layer architecture can simultaneously enhance the performance of diverse applications, we repeated our simulations with the TCP and UDP applications executing in parallel [10]. The multi-service link layer approach provided similar gains to those in single application tests. FTP performance improved by 11-2425%, depending on the topology and native loss rate, while conferencing delay was kept low since the scheduler prevented the TCP data stream from stealing UDP bandwidth.

IV. Best-effort Service Interface

A critical component of a multi-service link layer is a function for mapping IP packets to available services. The lowest common protocol layer on the Internet, the network layer, provides only a single, best-effort, packet delivery service. The assumption is that higher layer protocols can extend this service to satisfy additional application requirements. Our simulations show however that multiple link layer services are needed to enhance Internet application performance over wireless links. Since IP and the protocols using it are not aware of multi-service links though, they cannot map their requirements to available services.

In order to remain compatible with the existing Internet infrastructure, our architecture should perform this mapping without changing the interface between the link and network layers. Due to the scale of the Internet, such changes would take years to propagate everywhere. Performance should always be at least as good as with a single service, i.e. enhancing the performance of some applications should not degrade the performance of others. Finally, applications should never be mapped to services that degrade their performance. These requirements ease deployment as they allow services to be introduced gradually, to selectively enhance performance for some traffic, without affecting the rest.

Our link layer has single entry and exit points to ease its integration with existing protocol stacks. Since we cannot change this interface, the only information we can use for service selection is the incoming IP packets themselves. In order to match application requirements with available services we can use a classifier which employs heuristic rules to recognize certain applications based on IP, TCP and UDP header fields. All other traffic uses the default (native) link service. Figure 4 shows data flow in this heuristic classifier for IPv4. IP packet headers are masked to isolate the fields needed 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. Unknown applications are mapped to entries pointing at the default service, which provides the same performance as with a single service link layer.

Whether higher layers perform packet scheduling or not, they expect the link layer to allocate link bandwidth as if only a single service was available. For classification to be transparent, the services sharing the link should respect this (implicit) bandwidth allocation, even though each service may add arbitrary amounts of recovery overhead. In our architecture, after packets are assigned to services the classifier uses their size to implicitly deduce the share of the link allocated to each service. Over an implementation dependent interval, the classifier divides the amount of data assigned to each service 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 used in the rate table employed by the frame scheduler. Thus, unknown applications are allocated at least the same amount of bandwidth as in a single service scheme, while known applications 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 service properties.

Heuristic packet classification is based on the Protocol field of the IP header, which indicates TCP, UDP, or another protocol. All TCP applications can be mapped to a single TCP enhancement service, implemented by one of the link layer schemes mentioned above. UDP applications have very diverse requirements though, so 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 servers to be located at remote hosts. Thus, the protocol field along with well known ports can be used to recognize applications and map their data to the most appropriate services. 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. The same holds for IPv6, where the *Traffic Class* field is also being redefined to support Differentiated Services.

A significant drawback of heuristic classifiers is the amount of effort required to construct them. Applications and services must be manually matched for each type of link whenever new

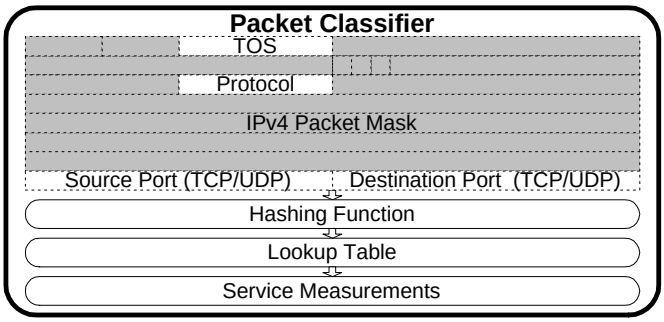

Fig. 4. Heuristic packet classifier

applications or services are added. Many applications will not be recognized, not only due to the large amount of applications in existence and the constant appearance of new ones, but also because many applications do not use well known ports at all. Although some QoS provisioning approaches combine the transport protocol, source/destination port and host address fields to classify packets [16], maintaining state for all these combinations requires end-to-end signaling and considerable storage, both of which are not available at the link layer. A more important problem for 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 [6], leaving only the TOS field visible, which is currently inadequate to describe application requirements.

V. Differentiated Services Interface

The single best-effort service offered by IP is reaching its limits 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 offered by statistical multiplexing, some type of performance guarantees must be introduced on the Internet. Users would presumably pay more for better Internet service if it would be cheaper to use the Internet rather than a circuit switched network for the same task. The main problem with Internet QoS is generally considered to be congestion. Applications may not be able to get the throughput they need due to contention for limited link resources and their end-to-end delay may increase due to queueing delays at congested routers. In addition, when router queues overflow, data loss occurs. UDP applications have to deal with these losses themselves, while TCP applications face additional delays due to end-to-end TCP recovery.

One approach for Internet QoS provision is the *Integrated Services* architecture [17], which has been criticized in two ways. First, it must be deployed over large parts of the Internet to be useful, since its guarantees rely on actions at every router on a path. Second, it requires resource reservations on a per flow basis, where a flow is defined as a stream of data between two user processes with the same QoS requirements. Since a huge number of flows exists, the scalability of any QoS scheme based on per flow state is limited. 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 problem is aggregation of flow state, but it is unclear how this can be achieved.

An alternative approach that aims to avoid these limitations is the *Differentiated Services* architecture [14]. In this scheme 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, which are either 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. This is denoted by the 8-bit Differentiated Services (DS) field of the IP header, which subsumes both the IPv4 TOS and the IPv6 Traffic Class field. 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 schemes. As an example, the *expedited forwarding* PHB provides a minimum amount of bandwidth at each router, for traffic that is 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 a 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 drop first the packets with highest drop preference. Flows are marked with higher drop preference levels when they exceed the traffic profile for 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 PHBs available and are meant to provide generic QoS levels, not application specific guarantees, hence the mapping of traffic classes instead of flows to PHBs. Only network entry points must be aware of both application requirements and PHB semantics to perform flow aggregation. Similarly, only domain entry points must be aware of the semantics of PHBs available in their neighboring domains to perform appropriate translations. Traffic policing, shaping and marking, is also only per-

formed at these points. For neighboring domains these 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 Integrated Services approach with its flow specific signaling.

Differentiated Services and multi-service link layers are solutions to orthogonal but complementary problems. Differentiated Services are concerned with congestion and its impact on throughput, delay and loss. The services provided are based on link independent PHBs, supported by packet scheduling and queue management mechanisms at the IP level. 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 a common set of link independent PHBs. The frame scheduling provided only protects services from each other by mirroring higher layer allocations, it does not provide end-to-end guarantees. By only providing Differentiated Services over wireless links we may offer to applications a nominal IP level QoS, but their actual performance will be limited by link losses. For example, even if we reserve wireless link bandwidth for a TCP traffic class, only a small fraction of it will be used due to losses and inefficient TCP error recovery. Multi-service link layers provide adequate recovery to fully utilize wireless links, but they also need higher layer guidance to perform scheduling.

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 layer schemes) 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 bit of the DS field. Each subclass would be mapped to an appropriate link layer service. All DS bits are only 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. 5 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 Traffic Class field of IPv6 headers.

This 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 previous applications. More importantly, the DS field is not hidden 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 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 traffic classes, 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 traffic class have separate bandwidth allocations at the IP level, then the bandwidth for all subclasses mapped to the same service is added to set its service rate. If subclasses share a common bandwidth pool within each class, then a service measurements module is inserted after the lookup table, as in Fig. 4, to automatically determine service rates.

VI. Advanced Quality of Service Interface

The performance of the 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 environmental conditions. If a characterization of the end-to-end performance of a network path is provided, applications can verify that a given service is suitable for their needs [18]. In addition, when path characteristics change significantly, fresh characterizations enable adaptive higher layers to modify their policies. In a hierarchical cellular system,[5] horizontal handoffs may cause error behavior to change, while vertical handoffs alter all wireless link characteristics. To describe the services offered over a network path, we must dynamically discover what 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. An end-to-end QoS module will thus be able to compose link metrics into end-to-end path metrics.

We have defined three link independent metrics reflecting the possible trade-offs in 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, as with 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 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 error recovery. It is greater than zero

[5]These are composed of multiple overlaid cellular systems: satellites cover the whole globe, cellular systems cover populated areas, and indoor WLANs cover buildings. Thus, in addition to horizontal handoffs between adjacent cells of the same system, the user can perform vertical handoffs between different systems.

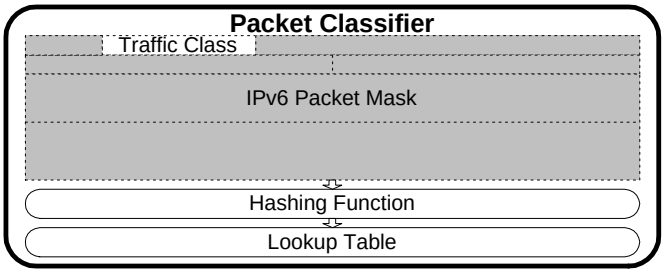

Fig. 5. Differentiated Services packet classifier

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 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 add the interval between loss of a recovered frame and its reconstruction from parity data to its one way delay estimate.

Delay 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 can 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 for data actually received, after discounting 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.

These metrics may be used at multiple layers to serve different needs, as shown in Fig. 6. The physical layer provides hardware dependent information, such as fixed one way delay, that may be used by the link layer services to provide their link independent g_i, l_i and d_i metrics. At the network layer, scheduling mechanisms such as CBQ may use those metrics to set bandwidth allocations for each service. End-to-end QoS schemes may use RSVP to gather information about node services in order to estimate end-to-end characteristics. Path characteristics can 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 encoding schemes that can deal with the residual loss rate of the path. Service metrics can be refined at each layer, as in the network layer where local metrics are used to compose end-to-end metrics.

To help higher layers deal with mobility, we can extend this interface to provide mobility hints to interested parties via upcalls, as shown in Fig. 6. The link layer can use hardware signals and combine them with its own state to flag events such as 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 [19]. 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 the 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.

This QoS interface was designed to fit the needs of *One Pass With Advertising* (OPWA) [18] resource reservation mechanisms. One pass schemes cannot specify a desired service in advance as they do not know what is available on a path [16], 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 [20]. Such reservations may fail due to tight restrictions on the first pass. In the 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 throughput, delay and loss restrictions imposed by wireless links, by looking at local multi-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, in order to satisfy the needs of adaptive mobility aware applications that can adapt their operation based on resource availability.

A related proposal was made in the context of IP mobility: a router could notify hosts communicating with a wireless host of new link characteristics after a handoff [21]. This approach however does not support link characterization between handoffs or multiple services. Another related proposal is an

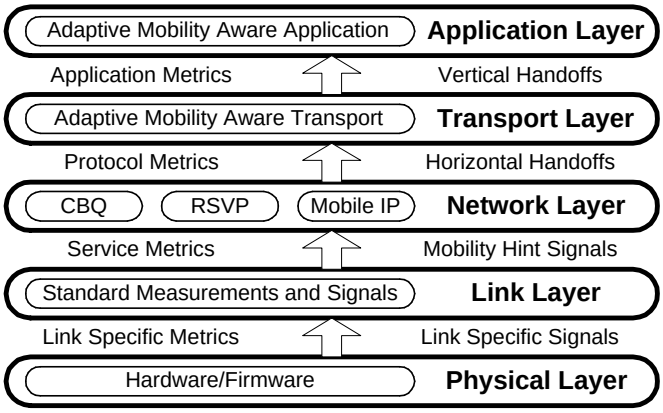

Fig. 6. Propagation of service measurement and mobility feedback

adaptation interface for mobility aware applications that notifies applications when their available bandwidth deviates from a specified range [22]. An application supporting multiple encodings of its data stream selects a bandwidth range for each, and switches encodings whenever the available bandwidth moves 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 on top of our mobility hints. A QoS management module at each host would accept bandwidth range requests from local applications. Instead of constantly monitoring the link, it would only check its data base after receiving a mobility notification, in turn notifying the applications whose ranges had changed.

VII. Conclusions

Even though extending the Internet over wireless links is rather 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 flow. 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 well-known 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 introducing recovery overhead. Finally, multi-service link layers can support an advanced QoS application interface, offering dynamic service discovery and synthesis.

References

[1] Stevens, W.: TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001 (1997)
[2] Nguyen, G.T., Katz, R.H., Noble, B., Satyanarayanan, M.: A trace-based approach for modeling wireless channel behavior. Proc. of the Winter Simulation Conference (1996)
[3] Karn, P.: The Qualcomm CDMA digital cellular system. Proc. of the USENIX Mobile and Location-Independent Computing Symposium (1993) 35–39
[4] Balakrishnan, H., Padmanabhan, V.N., Seshan, S., Katz, R.H.: A comparison of mechanisms for improving TCP performance over wireless links. Proc. of the ACM SIGCOMM (1996) 256–267
[5] Badrinath, B.R., Bakre, A., Imielinski, T., Marantz, R.: Handling mobile clients: A case for indirect interaction. Proc. of the 4th Workshop on Workstation Operating Systems (1993) 91–97
[6] Kent, S., Atkinson, R.: IP encapsulating security payload (ESP). RFC 2406 (1998)
[7] DeSimone, A., Chuah, M.C., Yue, O.: Throughput performance of transport-layer protocols over wireless LANs. Proc. of the IEEE GLOBECOM (1993) 542–549
[8] Xylomenos, G., Polyzos, G.C.: Internet protocol performance over networks with wireless links. IEEE Network **13** (1999) 55–63
[9] UCB/LBNL/VINT Network Simulator – ns (version 2). Source code available at `http://www-mash.cs.berkeley.edu/ns`
[10] Xylomenos, G.: Multi Service Link Layers: An Approach to Enhancing Internet Performance over Wireless Links. Ph.D. Dissertation, Dept. of Computer Science and Engineering, University of California, San Diego (1999)
[11] Nanda, S., Goodman, D.J., Timor, U.: Performance of PRMA: a packet voice protocol for cellular systems. IEEE Transactions on Vehicular Technology **40** (1991) 584–598
[12] Brady, P.T.: Evaluation of multireject, selective reject, and other protocol enhancements. IEEE Transactions on Communications **35** (1987) 659–666
[13] Xylomenos, G., Polyzos, G.C.: Link Layer Support for Quality of Service on Wireless Internet Links. IEEE Personal Communications **6** (1999) 52–60
[14] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An architecture for Differentiated Services. RFC 2475 (1998)
[15] Golestani, S.: A self-clocked fair queueing scheme for broadband applications. Proc. of the IEEE INFOCOM (1994) 636–646
[16] Zhang, L., Deering, S., Estrin, D., Shenker, S., Zappala, D.: RSVP: A new resource reservation protocol. IEEE Network **7** (1993) 8–18
[17] Clark, D., Shenker, S., Zhang, L.: Supporting real-time applications in an integrated services packet network: architecture and mechanism. Proc. of the ACM SIGCOMM (1996) 243–254
[18] Shenker, S., Breslau, L.: Two issues in reservation establishment. Proc. of the ACM SIGCOMM (1995) 14–26
[19] Perkins, C.: IP mobility support. RFC 2002 (1996)
[20] Ferrari, D., Verma, D.C.: A scheme for real-time channel establishment in wide-area networks. IEEE Journal on Selected Areas in Communications **8** (1990) 368–379
[21] Johnson, D.B., Maltz, D.A.: Protocols for adaptive wireless and mobile networking. IEEE Personal Communications **3** (1996) 34–42
[22] Noble, B, Price, M., Satyanarayanan, M.: A programming interface for application-aware adaptation in mobile computing. Proc. of the 2nd USENIX Symposium on Mobile and Location-Independent Computing (1995) 57–66