UNIVERSITY OF CALIFORNIA, SAN DIEGO

Multi Service Link Layers:
An Approach to Enhancing Internet Performance over Wireless Links

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Computer Science

by

George Xylomenos

Committee in charge:

Professor George C. Polyzos, Chair
Professor Rene L. Cruz
Professor Joseph C. Pasquale
Professor Ramesh R. Rao
Professor Bennet S. Yee

1999

The dissertation of George Xylomenos is approved, and it is acceptable in quality and form for publication on microfilm:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

1999

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

As a doctoral thesis is the culmination of many years of learning, I think it is only appropriate to begin by thanking my teachers. I am grateful to the members of my doctoral committee for their time, support, objections and suggestions. I was very fortunate to find five people that really understood my work. A special thanks is due to my thesis advisor, G. Polyzos, for guidance, encouragement and support during all my years at graduate school. Our joint work on many research topics beyond my thesis, as reflected in numerous publications, witnesses his influence on my academic development. His guidance and support was also invaluable in terms of professional development. For all his contributions to my work and my personality, I am very grateful. My advisor was also instrumental in obtaining financial support for my studies and research. He and J. Pasquale provided the excellent infrastructure of the Computer Systems Laboratory, a great environment to work in.

Among my other teachers, I have only space to single out a few more for their special contributions. C. Papadimitriou tremendously helped me getting into graduate school and was a constant source of support. E. Giakoumakis walked me through the agonizing process of writing my first publication quality paper. J. Cavouras was a huge overall influence on my academic choices and beyond. Many thanks also to R. Cruz, J. Pasquale, R. Rao, M. Bekakos, E. Magirou and T. Kalamboukis for their very warm reference letters. Going many years back, I am grateful to my uncle P. Valakitsis for first introducing me to computing and encouraging me to start programming. Very appropriately, this list ends with Vassilis and Argyro Xylomenos, my parents and my first teachers, inside and outside of a classroom. Their help, support and love was, and still is, invaluable.

At a personal level, many friends have provided moral support through my graduate school years away from my home country. Thanks to all my roommates that respected my unusual working habits and provided a great environment at home. I met numerous great people in person or through electronic discussion lists that helped me remain in touch with the outside world. Most of all however, I want to thank all my friends in Greece for keeping in touch with me and supporting my choices despite my long absence. After all those years, I still feel that my home is there, thanks to my friends and family.

VITA

May 29, 1971	Born, Athens, Greece
1993	B.S., Athens University of Economics, Athens, Greece
1996	M.S., University of California, San Diego
1997	C.Phil., University of California, San Diego
1999	Ph.D., University of California, San Diego

PUBLICATIONS

George Xylomenos and George C. Polyzos, "Internet Protocol Performance over Networks with Wireless Links," *IEEE Network*, 1999.

George Xylomenos and George C. Polyzos, "TCP and UDP Performance over a Wireless LAN," *Proceedings of the IEEE INFOCOM '99*, pp. 439-446, 1999.

George C. Polyzos and George Xylomenos, "Enhancing Wireless Internet Links for Multimedia Services," *Proceedings of the International Workshop on Mobile Multimedia Communications (MoMuC) '98*, pp. 379-384, 1998.

George Xylomenos and George C. Polyzos, "Enhancing Wireless Internet Links," *Proceedings of the 1998 International Conference on Telecommunications*, pp. 394-398, 1998.

George Xylomenos and George C. Polyzos, "IP Multicast Group Management for Point-to-Point Local Distribution," *Computer Communications*, Vol. 31, No. 18, pp. 1645–1654, 1998.

Joseph C. Pasquale, George C. Polyzos and George Xylomenos, "The Multimedia Multicasting Problem," *ACM Multimedia Systems*, Vol. 6, No. 1, pp. 43-69, 1998.

George Xylomenos and George C. Polyzos, "IP Multicasting for Point-to-Point Local Distribution," *Proceedings of the IEEE INFOCOM '97*, pp. 1382–1389, 1997.

George Xylomenos and George C. Polyzos, "IP Multicast for Mobile Hosts,", *IEEE Communications Magazine*, Vol. 35, No. 1, pp. 54–58, 1997.

FIELDS OF STUDY

Major Field: Computer Science

Studies in Computer Systems.
Professor George C. Polyzos, University of California, San Diego.

Studies in Communication Networks.
Professor George C. Polyzos, University of California, San Diego.

ABSTRACT OF THE DISSERTATION

Multi Service Link Layers:
An Approach to Enhancing Internet Performance over Wireless Links

by

George Xylomenos
Doctor of Philosophy in Computer Science
University of California, San Diego, 1999
Professor George C. Polyzos, Chair

Wireless communications and Internet use have both experienced explosive growth rates during the 1990's. Unfortunately, the performance of Internet applications over wireless links is severely degraded by transmission errors. Previous approaches to those performance problems have not succeeded in bridging the gap between end-to-end application requirements and local wireless link quality so as to offer a universal solution.

Aiming to solve wireless problems at their source, we concentrated on the link layer of the network protocol stack. To understand what types of solutions can optimize Internet performance over wireless links, we performed an extensive simulation study of different applications over a variety of transport layer protocols, wireless links, network topologies and link layer mechanisms. Our analysis reveals that the best solution for each situation depends on underlying link properties but not on awareness of higher layer semantics. We found that the use of appropriate link layer error control mechanisms leads to tremendous improvements in application performance, for both loss intolerant and loss tolerant applications. However, different solutions are preferable to satisfy diverse application requirements.

In order to optimize the performance of heterogeneous mixes of Internet traffic over wireless links, we proposed a *multi service link layer* architecture that allows the simultaneous operation of multiple link layer mechanisms, each satisfying the needs of a particular class of applications. Our proposal can be extended to support future needs and customized for any wireless link with minimal effort. We evaluated this architecture by repeating our extensive simulations with all applications operating simultaneously over multi service link layers, using

appropriate mechanisms for each type of traffic. Our results show that all applications improved their performance by virtually the same factor as when operating in isolation over their preferred link layer mechanisms.

Our architecture can be transparently integrated with the existing Internet to enhance application performance without any modifications to other protocols. It can also complement emerging Quality of Service provisioning schemes so as to combine congestion and error management. Finally, it provides standardized performance metrics that can be used to dynamically create end-to-end services for adaptive protocols and applications.

Chapter 1

Introduction

This chapter serves as a general introduction to the dissertation. Section 1.1 identifies the key factors that motivated our research. Section 1.2 briefly presents the problems that we studied and our research contributions. Section 1.3 is a guide to the structure and organization of the remainder of the dissertation.

1.1 Motivation

Since its inception and throughout its evolution, the worldwide Internet has extended its reach to new communication technologies and systems not long after each became available. It is not surprising then that wireless links, such as satellite and terrestrial microwave ones, have long been a part of the Internet, or that emerging digital wireless systems have generated a lively interest with respect to their seamless integration into the Internet. This ubiquity is largely owed to the communication technology independent design of IP (*Internet Protocol*), the (inter)network layer protocol of the Internet. IP offers a common interface to higher layer protocols over a wide range of communication links, aggregating dissimilar interconnected networks into a global entity. Application oriented protocols built on top of IP have enabled the development and widespread use of applications ranging from file transfer and electronic mail to the World Wide Web and audio/video conferencing.

The explosive growth of the Internet in recent years can only be paralleled by the similarly astonishing growth in the popularity of wireless communications as digital *Cellular Telephony* has spread worldwide with tens of millions of subscribers [1, 2]. It is unavoidable to

consider the implications of combining these two technologies: ubiquitous network connectivity without wiring or location constraints. In the local area this is becoming practical with the emergence of low cost *Wireless Local Area Networks*. Both technologies are evolving rapidly, and next generation systems promise improved bandwidth capacities, reduced cost and enhanced interoperability between systems from different vendors. Other emerging technologies such as digital *Cordless Telephony* and *Low Earth Orbit Satellites* will further expand the available wireless connectivity options [3] and, eventually, the reach of the Internet.

Even though wireless links are not new to the Internet, the bulk of its traffic has been carried in the past by wired links which have traditionally been the focus of Internet protocol development. With the shift to optical fiber in the past decade, this has meant a concentration on links with decreasing error rate and increasing bandwidth. Wireless systems however consistently lag behind wired ones, due to both physical and economic factors, generally exhibiting higher error rates and lower bandwidth capacities. As a result, protocol design assumptions that were reasonable for wired links are proving inappropriate for wireless links, leading to performance degradations that are far more serious than the actual error rates would imply. For example, a 2% IP packet error rate can cause application throughput over a wireless LAN to drop by half [4]. Digital cellular systems exhibit even higher error rates and worse application performance [5]. These shortcomings significantly hinder user acceptance of wireless Internet links.

Due to its popularity, the Internet is increasingly being used by applications not anticipated in its original design. For example, voice transmission over the Internet exploits the reduced communication cost due to statistical multiplexing but must also deal with unpredictable throughput and delays. As these applications become more divergent, the Internet will have to evolve in order to provide various *Quality of Service* (QoS) levels depending on application requirements, by, for example, offering maximum delay or minimum throughput guarantees. Supporting enhanced services over wireless links is even more challenging than improving the, already problematic, performance of conventional applications. Given the inherent shortcomings of wireless links, any improvements in their performance reflect specific tradeoffs between the individual characteristics of a link, such as loss rate, delay and throughput. Applications with different requirements may thus favor different tradeoffs depending on their needs.

This dissertation examines the issue of how best to improve Internet application perfor-

mance over various types of digital wireless links, from the viewpoints of different applications and their diverse QoS requirements. We identify the requirements of different Internet applications and measure their performance over multiple types of wireless links. We describe various link layer error recovery mechanisms and measure their performance benefits for each application and link type and conclude that *no single mechanism fits all situations*. We proceed by showing how multiple link layer mechanisms can be combined in a single *multi service link layer* that can be extended to support additional applications and wireless link types in a scalable, easy to program and efficient manner. Our measurements show that multiple applications can simultaneously optimize their performance using our scheme. We finally show how multi service link layers can interact and exchange information with higher layers in the framework of a QoS aware network protocol stack, so as to provide support for diverse application oriented services.

1.2 Scope and Contributions

The research effort presented in this dissertation consists of two complementary tracks. The first track deals with improvements to individual application performance over wireless links. The second track deals with multi application support and its interaction with QoS provisioning at higher layers. We have performed a comprehensive analysis of Internet application performance over wireless links, by extensively simulating numerous combinations of applications, transport protocols, wireless links, network topologies and link layer mechanisms. Our measurements show that the most profitable approach for each situation depends on individual application requirements and link characteristics. In order to enhance performance for arbitrary mixes of applications and links, we have proposed a multi service link layer architecture that can be easily extended to support additional applications and wireless links. Extensive simulations show that this scheme can simultaneously optimize the performance of diverse applications by employing multiple independent link layer mechanisms. We also show how this architecture can be integrated with higher layers to support multiple QoS levels. Thus, our research on local QoS complements research on end-to-end QoS provision and can be used as the basis for extending advanced Internet services over wireless links.

To further clarify the research scope and contributions of the dissertation, we give a

brief overview of the issues we examined and the conclusions we have reached.

- *Where in the protocol stack should we concentrate?* Previous research showed that degraded Internet performance over wireless is mostly due to mismatches between higher layer expectations and wireless link quality (Chapter 2). We could thus either modify higher layers or provide enhanced link layer schemes. We argue that link layer solutions are applicable to any link and network topology and are easier to deploy than higher layer modifications (Chapter 2). Our measurements show that application performance can be greatly enhanced without *any* higher layer modifications (Chapters 4 and 5).

- *Does a single link layer work for all applications?* Previous research has concentrated on enhancing the performance of *some* TCP applications only. We show that link layer schemes previously considered appropriate for any TCP application are in fact inappropriate for the most popular one (Chapter 4). We then describe a UDP application with strict QoS requirements, and show that its performance is optimized by a novel scheme that we designed, which is fundamentally different than those suited to TCP (Chapter 5).

- *Do we need to violate layering to optimize performance?* Some researchers have argued that wireless link layers *must* be aware of higher layer semantics in order to optimize performance. We show that those arguments were flawed. Our measurements show instead that link layer schemes can optimize performance without *any* awareness of higher layer *or* application semantics (Chapters 4 and 5).

- *Does the same link layer work for all links?* Previous proposals have only been tested under a single type of link and error model. Our measurements show that different approaches work best for different links and error conditions (Chapters 4 and 5). This implies that end-to-end modifications *cannot* match the performance improvements provided by customized link layer schemes over heterogeneous network paths.

- *How can multiple schemes be combined over a single link?* Existing link layers providing a single service are inherently unable to jointly optimize multiple applications. We therefore propose a *multi service link layer* architecture that combines multiple simple mechanisms to simultaneously support diverse requirements. Our scheme provides isolation of services and controlled sharing of the physical link in an extensible manner (Chapter 6).

- *How well does a multi service link layer perform?* We repeated our extensive single application tests with multiple simultaneous applications over the multi service link layer scheme, using appropriate services for each type of traffic. Our results show that with our scheme each application improves its performance by virtually the same factor as when operating in isolation over the same service (Chapter 7).

- *How does a multi service link layer interact with higher layers?* We present a simple heuristic scheme to match higher layer requirements with available link services so as to transparently improve performance in the existing Internet. We show how an emerging higher layer Quality of Service provisioning model can complement our scheme. We finally show how advanced end-to-end application oriented services may be composed dynamically by selecting appropriate link layer services (Chapter 8).

1.3 Structure of Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides background information on wireless links and Internet protocols and explains the reasons for their degraded performance. We present existing approaches to these problems and identify their limitations. We derive a set of general requirements for any universal performance enhancing approach and argue that they can best be met by suitably designed link layer schemes.

Chapter 3 describes our measurement setup. We first introduce the simulator we used and the different links and error models employed, including their parameters and simulator implementations. We then present the network topologies used. Throughout, we discuss the different aspects of the problem captured by our test suite, as well as its limitations.

Chapter 4 describes the TCP applications we employed and their simulator implementations. We then discuss the link layer mechanisms tested and their implementations. We present extensive performance measurements under a variety of single application scenarios. We compare and contrast our findings with previous research and select the most promising mechanisms for further testing.

Chapter 5 describes the UDP application we chose for testing and its simulator implementation. We introduce link layer mechanisms for real time UDP streams, including a novel modification to an existing scheme. We present extensive performance measurements showing

that the best approaches for UDP are different than those for TCP.

Chapter 6 discusses the design requirements for a solution combining the best schemes for TCP and UDP. We present an architecture for a *multi service link layer* that can provide multiple simultaneous services to higher layers. We show how this approach isolates services from each other and provides controlled link sharing via efficient frame scheduling.

Chapter 7 presents performance measurements with multiple applications sharing a variety of links using our multi service link layer scheme. These results closely approximate those obtained in single application scenarios, scaled for the reduced bandwidth due to link sharing. We conclude that our architecture is capable of jointly optimizing multiple applications.

Chapter 8 shows how our multi service link layer scheme can be integrated into the existing Internet to enhance performance without any modifications to higher layers. We then discuss how it can be combined with emerging higher layer QoS provisioning schemes to support both congestion and error control. We finally show how higher layers can exchange information with our link layer in order to dynamically compose end-to-end services.

Finally, Chapter 9 summarizes our results and identifies the original research contributions of this dissertation. We conclude with directions for further research on multi service link layers and their interaction with end-to-end Quality of Service provisioning.

Chapter 2

Background and Related Work

This chapter provides background material for the remainder of the dissertation, including a critical review of related research. Section 2.1 surveys digital wireless link characteristics while Section 2.2 discusses Internet protocol design concepts and the causes for their poor performance over wireless. Section 2.3 outlines the magnitude of these problems. We review existing approaches to enhancing Internet performance over wireless in Section 2.4. Drawing on their shortcomings we state a number of requirements for a universal approach in Section 2.5 and argue that only a link layer based approach can fulfill these requirements.

2.1 Wireless Link Characteristics

Our research focuses on digital *Cellular Telephony* (CT) and *Wireless LAN* (WLAN) systems using radio frequency (RF) modems. These systems are reasonably priced, widely available and targeted to end users rather than network operators. Both systems must share the spectrum with, occasionally malicious, external RF sources, as well as with neighboring systems of the same type. Their error rates are thus higher and their available bandwidth lower than those of the shielded and isolated wired links. Due to their relatively small coverage areas, signal propagation delays between communicating devices are small compared to those of geostationary satellites, hence transmission time usually dominates total delivery delay. Terrestrial obstructions such as buildings, furniture and people, cause both indoor and outdoor links to suffer from multipath fading. In addition, mobility, a key characteristic of cellular systems, constantly changes the fading and interference characteristics of a link. Therefore, the error behavior of

these systems varies in a faster and more unpredictable manner than that of satellite links.

On the other hand, these systems also have considerable differences. WLANs offer relatively high bandwidth shared among users within a limited coverage area, usually inside a building. Cellular systems offer low bandwidth circuit mode links over a larger area, with both indoor and outdoor coverage. Cellular systems are built to enable mobility and promote bandwidth sharing between adjacent coverage areas, a characteristic that has only recently appeared in WLANs. Cellular systems are designed for real time voice telephony, while WLANs are meant for data communications. Many of these distinctions are becoming increasingly blurred however as traditional voice networks are merging with the Internet, cellular systems evolve to higher capacities and variable bit rate allocations, and WLANs start supporting user mobility between neighboring systems. To assess the capabilities of these wireless links, we review the main characteristics of some existing systems in the following paragraphs.

2.1.1 Wireless LAN Systems

One of the earliest available WLANs was the Lucent (originally NCR) WaveLAN [6]. The original versions used radios in the 900 MHz unlicensed *Industrial, Scientific and Medical* (ISM) band, while later versions use the 2.4 GHz ISM band. The system supports a peak bandwidth of 2 Mbps, shared between all hosts within range of each other. The medium access scheme is CSMA/CA instead of CSMA/CD, as used in Ethernet. The reason is that *collision detection* (CD) is expensive in terms of bandwidth for the radios, so *collision avoidance* is employed instead. The service offered emulates an Ethernet LAN: frames have the same headers and *Cyclic Redundancy Codes* (CRCs) for error detection, frame size is up to 1500 bytes, and the service provided is connectionless best effort frame delivery. Although delivery delay is unpredictable due to the need for channel contention, in practice it is very low: propagation delay is up to a few milliseconds due to a relatively short range (300-1000 ft, depending on the environment), while frame transmission time is kept short by the high link speed.

WaveLAN performance has been widely studied and reported in the literature. The system is robust in the presence of narrowband interference and obstacles within its operating range [7]. Bit error rates are less than 10^{-5} according to system specifications [8]. Typical frame error rates are less than 2.5% using 1500 byte frames. Human bodies can also hinder transmission, a likely event in an office environment. Interference problems are caused by spread

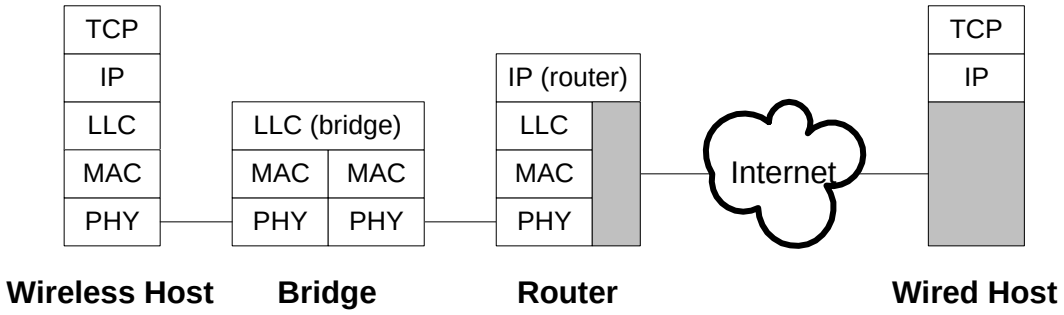

Figure 2.1: Connectivity between a wireless LAN and the Internet

spectrum devices operating at similar frequencies, such as cordless telephones [9], and other WaveLAN transmitters belonging to neighboring networks. A threshold mechanism is supported that can isolate WLANs that are sufficiently separated in space by ignoring transmissions whose signal strength is below the threshold. On the other hand, the system does not offer either power control or multiple base frequencies, so adjacent networks have to share the available spectrum. Implementation differences between desktop and laptop network interfaces cause a throughput asymmetry between them [10]. Our own measurements indicate that host processing power affects throughput and frame loss between heterogeneous systems [11].

To achieve interoperability between WLANs from different vendors the IEEE 802.11 standard was designed [12], specifying a system similar to, but more advanced than, WaveLAN. This standard is embodied in products such as the WaveLAN IEEE and WaveLAN IEEE Turbo, Lucent's 2 Mbps and 10 Mbps implementations of IEEE 802.11, respectively [8]. New features supported by IEEE 802.11 include RTS/CTS (request/clear to send) messages exchanged between sender and receiver before data transmission, to ensure that the neighbors of both peers will remain silent in the next transmission interval; contention free acknowledgments for successfully received data frames, in the absence of which the sender retransmits the unacknowledged frame; and an operating mode where a single host acts as a master offering centralized access control and contention free data transmission.

Connecting a WaveLAN to the Internet can be achieved in the same manner as for a conventional Ethernet. One option is to use a bridge to transparently attach the wireless LAN to a wired one, as shown in Figure 2.1. The bridge contains medium access control (MAC) and

physical layer (PHY) functionality for both networks. It copies frames between the two at the logical link control layer (LLC). As far as higher layers are concerned, a single LAN exists. The existing router for the wired LAN can therefore be directly used by the wireless hosts as well. The other option is to eliminate the bridge in Figure 2.1 by directly attaching a wireless interface to an IP router. Although the wireless hosts use the router in the same manner as in the bridged configuration, in this scheme the wireless network is a separate entity to higher layers.

2.1.2 Digital Cellular Systems

Current digital cellular systems are characterized by modest transmission bandwidths, small frame sizes, and circuit mode operation, due to their design for voice telephony. Data services are provided by including in each frame data handed to the link layer by higher layers rather than by a voice encoder. The lower transmission rate and longer range of cellular systems compared to WLANs lead to higher total delays. The outdoor cellular environment is very harsh with interference and multipath fading caused by buildings and hills. To optimize voice performance, cellular systems use short frames that suffer from losses of 1-2% [5]. As long as these losses are random, the voice encoder/decoder can avoid audible quality degradation.

Due to the real time nature of voice telephony which requires isochronous frame delivery, only *forward error correction* (FEC) schemes are employed to achieve these frame error rates. The error process at the physical layer however usually produces long bursts of bit errors that may span multiple frames, so individually encoding each frame does not provide sufficient recovery. To randomize bit errors multiple frames are interleaved, i.e. their bits are reordered before transmission, with the reverse process taking place at the receiver. With this technique error bursts are spread across multiple frames, allowing their embedded FEC scheme to recover most of them *and* making frame errors more random. However this scheme requires multiple frames to be received before encoding and decoding take place, considerably increasing total delay. For one such system for example, the one way delay is 100 ms [13].

For many data applications even these loss rates are unacceptable, necessitating additional error recovery. For this reason, most cellular systems offer a *non-transparent mode* for data in addition to their native *transparent mode* [14]. The non-transparent mode is designed for non real time applications such as telefascimile (FAX) that expect a virtually error free channel. It may use any type of mechanism to improve the link, including *automatic repeat request* (ARQ)

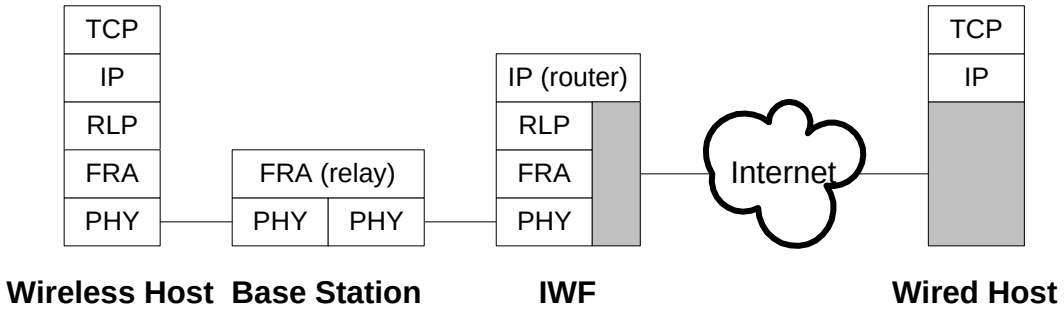

Figure 2.2: Connectivity between a digital cellular system and the Internet

for frame retransmissions or additional FEC. These link layer protocols, commonly called *Radio Link Protocols* (RLPs) may offer additional services on top of error recovery, such as segmentation and reassembly of variable size higher layer packets into fixed size link frames.

Cellular systems use an *Interworking Function* (IWF) to interface with other networks [14]. For example, communication with analog telephones or modems is provided by the IWF transforming the digital voice or data frames of the CT system to analog waveforms. To facilitate Internet access, an appropriate RLP may be introduced between a wireless host and the IWF as depicted in Figure 2.2. The IWF is located at the boundary between the cellular system and the Internet at the network provider's premises and includes an IP router. A wireless host communicates via a cellular link with the base station of its cell, which is in turn connected via a wired link to the IWF. A simple frame delivery service (FRA) is provided between the IWF and the wireless host for voice telephony. The RLP is used to improve reliability and to encapsulate IP datagrams into link layer frames between the IWF and the wireless host.

Current digital CT systems are based on either time division (GSM and IS-54) or code division (IS-95) multiplexing of the medium. The European standard GSM supports 9.6 Kbps full rate data channels at average bit error rates of 10^{-3}. Its non-transparent mode uses 240 bit frames and a Selective Repeat ARQ scheme [15] that reduces bit error rate to 10^{-8} at the expense of variable throughput and delays [14]. The North American IS-54 system also supports 9.6 Kbps full rate links but adds a considerably more sophisticated ARQ based RLP using 256 bit frames. This RLP can negatively acknowledge multiple outstanding frames at once by exploiting additional state at the sender [16], achieving an effective throughput of 7.8-8.2

Kbps, with variable delays due to retransmissions. The North American IS-95 system supports 8.6 Kbps full rate links using 172 bit frames. Its non-transparent mode first encapsulates variable sized higher layer packets into PPP frames [17] and then segments these into fixed size link frames [5]. This combines the convenience of variable sized packets with the simpler error recovery possible with fixed size frames. This RLP only retransmits lost frames a limited number of times before giving up, delegating further recovery to higher layers. The effective packet loss rate for 1000 byte packets is around 10^{-4} at user data rates of 7.5-8 Kbps [5]. More details on these and other link layer mechanisms are provided in Chapters 4 and 5.

2.1.3 Future Systems

In the near future a new generation of digital cellular systems will be introduced, exploiting new, and more generous, RF spectrum allocations. These systems, generally referred to as *Personal Communications Systems* (PCS), are expected to be more accommodating to Internet needs than their predecessors. An existing scheme that extends Internet style packet data access to wireless hosts is the *Cellular Digital Packet Data* (CDPD) system [18]. CDPD multiplexes data packets over unused IS-54 voice channels on a demand driven basis. An evolution of this approach is the *General Packet Radio Service* (GPRS) scheme that is being introduced to GSM [19]. GPRS allows multiple GSM time slots to be allocated to a single user in each frame, thus supporting peak user data rates of up to eight times the basic GSM data rate. A similar packet data scheme allowing multiple channels to be assigned to a user is planned for next generation Wideband CDMA based systems, along with vastly increased data rates, ranging from 384 Kbps for wide area coverage to 2Mbps for local area coverage [20].

Digital cordless telephony lies between digital cellular and wireless LANs, combining small coverage areas and high available bandwidth with circuit mode links. Emerging standards will allow equipment from different vendors to interoperate, enabling handheld devices to migrate between base stations, with each base station forming its own *picocell*. The European DECT system offers 32 Kbps data links and is explicitly designed for picocellular applications by offering authorization support and mobility management [21]. The cellular concept is extended in another direction by constellations of low earth orbit satellites, such as those of the Iridium system [22], offering connectivity even in areas without any terrestrial infrastructure. The area covered by each satellite forms a *macrocell*, which moves as the satellite orbits the

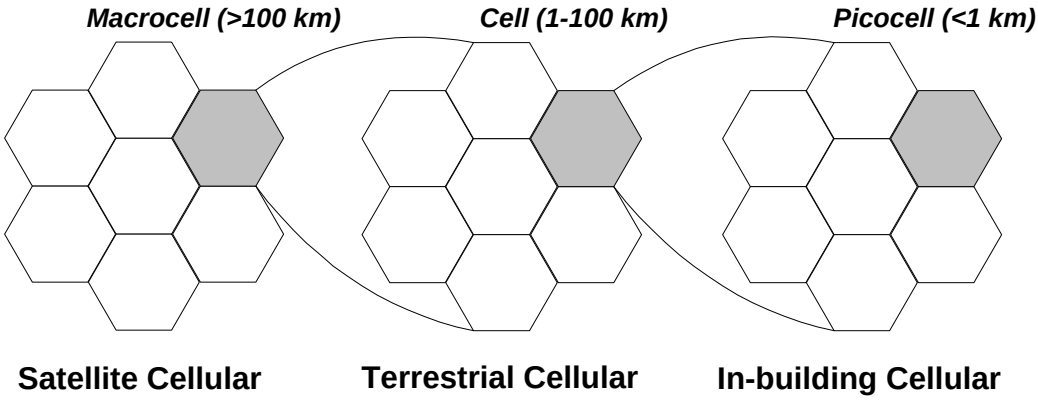

Figure 2.3: Hierarchical cellular system

earth.

Each of those cellular systems will most likely retain its appeal due to economic considerations. Picocells require a dense mesh of base stations, whose cost can only be justified within buildings. Cellular systems with standard sized cells will provide lower bandwidth coverage in well populated areas, while sparsely populated areas that do not warrant the cost of terrestrial cellular infrastructure will be covered by satellites. This will give rise to a hierarchical cell structure [23], as depicted in Figure 2.3: higher level cells are overlaid in areas with more users by multiple lower level cells. Users can use the highest bandwidth system available in each location and move from cell to cell either within the same system (performing *horizontal handoffs*, as in existing cellular systems), or from one system to another (performing new style *vertical handoffs*), depending on coverage.

Since handoffs between cells momentarily disrupt connectivity in cellular systems, we expect hierarchical cellular systems to provide additional challenges in this respect. In the picocells, handoff frequency will increase due to the small cell radius. In addition, when handoffs become possible between different systems, connections will face two levels of link performance variability. Short term performance variations will be caused by environmental changes such as fading, in the same manner as today. Long term performance variations will also be caused by handoffs between different technologies (picocellular, cellular and macrocellular). These handoffs will dramatically change the performance parameters of the wireless part of end-to-end

paths, as each type of link will have its own characteristics.

2.2 Internet Protocol Characteristics

The “glue” that holds the Internet together is its network layer protocol, IP [24], which provides global host addressing and basic data delivery. IP was designed with the explicit goal of supporting the integration of heterogeneous networks into a global entity, hence its implementation requirements and service offerings are minimal. IP provides end-to-end delivery of variable sized messages, called *datagrams*, which may be reordered, duplicated or lost, without warning. To offer these best-effort IP services over a wireless, or any other, network, we only need to provide a link layer that encapsulates arbitrary IP datagrams into link frames and delivers them between adjacent nodes. Often, this is simply a matter of fragmentation and reassembly of datagrams into frames. Since IP does not enhance the service provided by individual links in any way, end-to-end performance is limited by the worst link on the path. Higher layer protocols may enhance this best-effort service to better support application requirements.

Transport layer protocols are the interface between user applications and the network, or, more accurately, IP. Even though they offer user oriented services, their design is based on assumptions about network characteristics. One choice offered on the Internet is UDP (*User Datagram Protocol*), essentially a thin layer over IP [25]. UDP provides a connectionless best effort message delivery service, without flow, congestion or error control. It only adds addressing of individual applications within a host on top of IP. This nearly direct access to IP is used by two broad categories of applications. First, applications for wired LANs, which can reasonably assume that the network is very reliable and has plenty of bandwidth available, so error and congestion control are not crucial. Second, wide area applications that have some real time requirements, such as audio/video conferencing. These applications usually prefer handling error and congestion recovery themselves so as to better meet their requirements.

TCP (*Transmission Control Protocol*) is the other common transport protocol choice offered on the Internet [26]. It provides a connection oriented reliable byte stream service that appears to applications similar to writing (reading) to (from) a sequential file. TCP also supports flow and congestion control, and segmentation and reassembly of the user data stream into/from IP datagrams. TCP data segments are acknowledged by the receiver strictly in order. When

arriving segments have a gap in their sequence, duplicate acknowledgments are generated for the last segment received in sequence. Losses are detected by the sender either by timing out while waiting for a transmitted segment to be acknowledged, or by a series of duplicate acknowledgments implying that the next segment in the sequence was lost in transit. Since IP may reorder datagrams, TCP cannot automatically assume that all gaps in the sequence mean loss, hence multiple duplicate acknowledgments must be received first. During periods of inactivity or when acknowledgments are lost, TCP detects losses by the expiration of timers. Since Internet routing is dynamic, a retransmission timeout value is continuously estimated based on the averaged round trip times of previous data/acknowledgment pairs. A good estimate is very important: large timeout values delay recovery after losses, while small values may cause timeouts to occur when acknowledgments are only delayed rather than lost [27].

Wired long haul links have been exhibiting decreasing error rates, due to the widespread use of optical fiber, while wired LANs are inherently very reliable. As a result, the statistical multiplexing of increasing traffic loads over the Internet has replaced bit errors with *congestion* as the dominant loss factor. Congestion occurs when routers are overloaded with traffic that causes their packet queues to build up, causing increased delays and eventually packet loss when queues are filled. When such losses occur, the best remedy is to reduce the offered load so as to drain router queues and restore traffic to its long term average rate [27]. A transport protocol therefore needs to adapt its maximum transmission rate, increasing it when the network has unused capacity, and decreasing it when congestion appears. Since congestion is the most common cause for loss on wired links, TCP assumes that *all* losses indicate congestion [27]. Hence losses cause, apart from retransmissions, the maximum transmission rate of TCP to be reduced and then gradually increased, probing the network for the highest acceptable load.

Figure 2.4 illustrates how TCP reacts to losses. TCP maintains an estimate of how much unacknowledged data can be outstanding in the network without causing congestion. The maximum amount of outstanding data is limited by the minimum of this *congestion window* (to ensure that routers are not overloaded) and the receiver's *advertised window* (to ensure that the receiver is not overwhelmed by the sender). Initially, the congestion window is set to one segment and a *slow start threshold* is set to a large value. While below the threshold, each new acknowledgment causes the congestion window to increase by one, thus doubling after each round trip time, i.e. an exponential increase. This is called the *slow start* phase. In the figure this

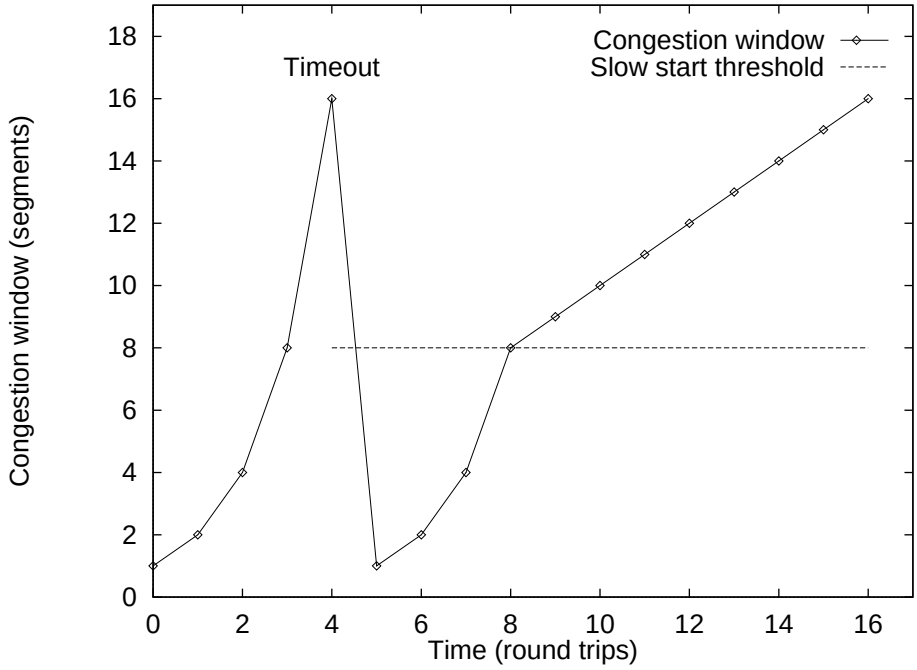

Figure 2.4: TCP congestion window behavior

increase stops after 4 round trip times when either a timeout or multiple (usually 3) duplicate acknowledgments indicate that a segment was lost. Immediately the slow start threshold is set to half the value of the congestion window, the congestion window is set to a single segment, and the lost segment is retransmitted. Slow start again takes place until the threshold is reached in 3 more round trip times, allowing congested routers to drain their queues in the meantime. Since congestion occurred at twice the current window size, from there on the congestion window increases by a single segment for each round trip time, i.e. a linear increase, instead of being doubled. This is called the *congestion avoidance* phase.

These mechanisms are part of the *TCP Tahoe* variant, the “baseline” TCP implementation [28]. The more advanced *TCP Reno* variant, very common in newer systems, differentiates between losses detected by timeouts, indicating that a large number of packets has been lost, and those detected by multiple duplicate acknowledgments, indicating that subsequent segments have been received. In the latter case, it is assumed that congestion is not very severe, so instead of slow start TCP Reno (roughly) restarts from the congestion avoidance phase, after recovering from the loss (by a retransmission) and halving the congestion window [28]. This reduces but

does not eliminate the significant drop in throughput during losses shown in Figure 2.4. When losses are clustered, the congestion window and threshold may be reduced repeatedly, forcing the congestion avoidance phase to start from decreasing window sizes. Even worse, when clustered losses cause another data segment to be lost during recovery, frequently the sender is prohibited by its reduced congestion window from sending enough additional data segments to trigger an adequate number of duplicate acknowledgments [29]. Hence, additional losses can only be detected by timeouts, which cause the even slower slow start phase to be entered.

2.3 Internet Protocol Performance over Wireless Links

Wireless Internet links invalidate the widely held assumption that the network only rarely, if ever, loses packets due to errors. A representative WLAN, the WaveLAN described above, when transmitting UDP packets with 1400 bytes of user payload over an 85 foot distance suffers from an average *frame error rate* (FER) of 1.55% [10]. These errors are clustered and their rate is mainly influenced by distance and obstacles between the hosts and frame size [10]. Reducing the frame size by 300 bytes cuts the error rate in half. Host mobility increases error rates for this WLAN by about 30%, in the absence of handoffs [10]. Wired LAN applications such as remote file access via the *Network File System* (NFS) [30], that in theory are suitable for a high speed WLAN, are effectively rendered unusable at these error rates. Most NFS implementations use 8 Kbyte UDP datagrams that are segmented into multiple fragments to fit the 1500 byte link frames. If one such fragment is lost, the whole datagram is wasted. The retransmission schemes in most NFS implementations are rudimentary, as errors are assumed to be very rare, with retransmission timeouts that can easily grow to several seconds. A user editing a remote file over a WLAN may end up waiting most of the time due to wireless errors.

TCP is inherently slower than UDP, not only due to its larger protocol headers, but also because reverse traffic (acknowledgments) must share the WLAN medium with forward (data) traffic. This can potentially cause collisions, which occur when both hosts start transmitting at the same time. In CSMA/CD wired LANs, collisions are detected very fast, causing transmissions to be aborted and restarted after a new contention period. In wireless LANs however collision detection is expensive to implement in terms of bandwidth, so collision avoidance is used instead. With CSMA/CA when the transmitter senses a silent medium it does not transmit

immediately as with CSMA/CD. Instead, it first waits for a random interval of time drawn from a *contention window*. If the medium remains idle after the interval has elapsed, transmission begins and the window is set to a small value, else the window is doubled and the transmitter must try again. When collisions occur, bandwidth is wasted while the frames are transmitted to completion, instead of aborted as with CSMA/CD.

Undetected CSMA/CA collisions are a problem since they are visible to higher layers. Our own measurements on the WaveLAN indicate that some implementations of the interface are subject to considerable collision rates with TCP, on the order of 10% with 1500 byte data segments. This causes TCP throughput to drop to 30% of its value in the absence of collisions [11]. Even without collisions, measurements of TCP file transfers over a WaveLAN using 1400 byte data segments revealed that the throughput achieved is reduced by 22% with a 1.55% frame error rate [10]. In both cases, these throughput reductions are due to TCP frequently invoking congestion avoidance mechanisms that reduce its transmission rate, even though the losses are not due to congestion. If errors were uniformly distributed rather than clustered in the latter case, throughput would drop only by 5.5% [10], since TCP performs worse with losses clustered within one transmission window, as explained above [29].

Cellular links are an order of magnitude worse than WLANs, since they suffer from 1-2% error rates [5] on frame sizes of 20-30 bytes. A full rate IS-95 link uses 172 bit frames, insufficient even for TCP and IP headers, so the link layer must segment IP datagrams into multiple frames. A 1000 byte TCP segment would be split in about 50 such frames. Assuming independent frame errors at a 1% rate, the probability that the packet will make it across the link is only 60%, while at a 2% rate it drops to 36% [5]. Reducing the datagram size reduces the datagram loss rate at the expense of increasing header overhead. However, TCP overhead can be reduced to 3-5 bytes per datagram by employing header compression, a technique appropriate for low bandwidth serial links [31]. This optimization is feasible *only* for the TCP/IP combination, not for UDP over IP. Note that cellular systems use separate uplink and downlink channels, so forward (data) and reverse (acknowledgment) traffic do not interfere as in the WLAN case.

When multiple wireless links are traversed in an end-to-end path, errors accumulate in a multiplicative manner. This occurs when users of distinct cellular or WLAN systems communicate via the wired infrastructure. TCP segments that make it across the first link successfully may have to be retransmitted over it if they are lost in the second one, wasting precious wire-

less bandwidth. Increased losses mean more frequent invocations of TCP congestion avoidance algorithms, hence even further reductions in throughput. In addition, recovery via end-to-end retransmissions becomes slower as the delay between the peers increases. The combined effect of mistaking wireless losses for congestion and slow end-to-end recovery is more pronounced on longer paths that require large TCP windows to keep data flowing: when the TCP transmission window is reduced after losses are detected, the end-to-end path remains underutilized until the window grows back to its appropriate size. This takes more time for longer paths since window growth is clocked by end-to-end TCP acknowledgments.

2.4 Existing Performance Enhancements

We have seen that the performance problems of Internet protocols and applications over wireless links are due to a mismatch between what the protocol or application expects and what the network actually offers. Although this is more pronounced with TCP, where the assumption that all errors are due to congestion is explicitly made, most UDP applications make similar assumptions. Traditionally, in the Internet model error recovery was delegated to end-to-end layers, to avoid duplication of effort, simplify link layer design, and avoid imposing error recovery overhead on applications that do not need it. Following this school of thought, one approach to improving higher layer performance is to modify higher layer protocols and applications to take into account wireless link shortcomings. An alternative viewpoint is that end-to-end layers cannot feasibly deal with every conceivable link limitation, hence a minimum level of quality must be provided by all links, and in particular wireless ones, to satisfy higher layer assumptions. We review both types of schemes in the following paragraphs.

2.4.1 Higher Layer Approaches

Most research on Internet protocol performance over wireless links has focused on TCP, since it is the most commonly used transport protocol on the Internet. TCP based applications experience significantly reduced throughput and increased delay due to the TCP assumption that all losses are due to congestion and its slow end-to-end recovery. In all wireless systems losses due to wireless errors are not uncommon. In cellular systems, during handoffs connectivity is temporarily lost and a timeout may be needed to initiate recovery. In addition,

timeouts can occur during handoffs even in the absence of losses. Since it is TCP assumptions that cause these problems, a direct remedy is to modify these assumptions. To avoid long pauses after handoffs, one approach is to initiate recovery right after a handoff completes, instead of waiting for a timeout [32]. This requires signaling from the layer handling mobility, usually IP with mobility extensions [33], to notify the transport layer of handoff completion. Invoking full congestion recovery procedures after every handoff still reduces throughput, so an alternative is to attempt to detect whether loss is due to mobility or congestion by exploiting mobility hints from lower layers [34]. With losses due to congestion, both slow start and congestion avoidance are used. With losses due to mobility, only slow start is used for faster recovery.

After a handoff, some congestion avoidance measures are required to probe the state of the link in the new cell. With losses due to errors however we should skip congestion avoidance completely. Since these losses are isolated to the local link, end-to-end retransmissions unnecessarily delay recovery. One way to improve error recovery is to *split* TCP connections using as *pivot points* those routers on the path connected to both wireless and wired links, for example the IP router in Figure 2.1 and the IWF in Figure 2.2. One TCP instance executes over each wired part while either another instance of TCP or another protocol executes over each wireless part [35, 36]. Each transport connection is independent, i.e. losses at a wireless link are only visible in its own connection and are recovered from locally. Software agents executing at each pivot point bridge these connections by copying data between them. When TCP is used over the wireless segments, it can recover fast due to the short paths involved. Alternatively, a transport protocol with faster error recovery mechanisms may be used [36].

The ideal scenario for these schemes is a path with one or two wireless endpoints, where the pivot points are the routers connecting the wireless links to the Internet. Handoffs in these schemes change the endpoint but not the pivot point, allowing the bridging agent to establish new connections and speed up recovery after a handoff. Completely independent connections have the drawback that TCP acknowledgments lose their end-to-end significance: instead of asserting that data were received at the destination, they mean that they were received by the pivot point. To avoid this problem, both data and acknowledgment streams may be bridged by the pivot agent. To speed up recovery in these cases the TCP instances executing over wireless links are modified by, for example, avoiding congestion control. Local TCP segment retransmissions have the side effect of inflating the TCP timeout estimates, hence slowing down its reaction

to congestion. A solution to this problem is to adjust the timestamps in retransmitted segments to account for the time spent during recovery [37].

To avoid wasted retransmissions during handoffs or periods of fading, a pivot agent can also choke the wired sender by closing the advertised window, as in M-TCP [38]. This causes the sender to go into *persist* mode, during which it periodically probes the receiver's window while freezing all pending timers. After connectivity is re-established, the advertised window is reopened and TCP can continue without any timeouts. Note however that shrinking the advertised window violates TCP implementation guidelines [26]. An alternative is to use an *Explicit Bad State Notification* (EBSN) to force the sender to keep resetting its timers during handoff or fade periods so as to avoid timeouts [39]. A complementary scheme to M-TCP for UDP based applications is M-UDP, which also masks losses due to handoffs or fades by retransmitting UDP messages when connectivity is re-established [40]. Since UDP applications are too diverse to be characterized by any common set of requirements, there has not been much research devoted to improving their performance. In fact, M-UDP is the only UDP enhancement scheme that we are aware of.

A claimed advantage of TCP modifications is that only end hosts, which are under user control, need to upgrade their software. Split connection schemes however require pivot points to be modified as well, and these are generally under network operator control. Some split connection schemes violate TCP semantics since acknowledgments lose their end-to-end significance, thus applications needing end-to-end reliability must use additional protocols above TCP. New transport protocols compatible with TCP are needed to maximize performance over wireless segments, since TCP is poorly suited to this task. The agents at the pivot points are complex: they must translate semantics and synchronize connections despite errors and handoffs. In addition, they must maintain separate state for each active TCP connection, a significant burden for routers handling numerous wireless hosts. They are also incompatible with IP security mechanisms that encrypt the datagram payload, completely hiding TCP headers [41]. Most of the schemes make assumptions about the location of the end hosts. The performance of split connection schemes is questionable for wireless links not at path ends or for multiple link wireless segments, where this approach degenerates to a link layer scheme.

2.4.2 Lower Layer Approaches

The alternative to end-to-end modifications at the transport or higher layers is to modify the link layers of wireless links so as to hide losses using local recovery mechanisms. Cellular systems offer non-transparent mode radio link protocols that enhance link reliability with this exact goal in mind [5, 14, 16]. These schemes are customized to specific links, hence they operate with detailed knowledge of physical layer properties such as error process characteristics, link states, frame sizes and hardware signals. They are generally unaware of the type of data they are carrying, hence they apply the same procedures for both TCP and UDP streams.

A link independent local recovery scheme, the *Snoop* protocol, adds link layer functionality to IP for local error recovery [42]. IP datagrams carrying TCP data are buffered at the base station before being transmitted towards a wireless host. They are retransmitted if they are not acknowledged by TCP within a short period of time or if duplicate TCP acknowledgments for previous data are received. The error recovery module at the base station *snoops* on all IP datagrams to gather TCP data and acknowledgment information. Buffered TCP segments that need to be retransmitted are inserted into the data stream transparently to the receivers. Similarly, duplicate TCP acknowledgments from the wireless host are silently suppressed by the base station when the missing segment can be locally retransmitted, hiding the loss from the sender at the other end of the path. By leveraging existing TCP messages this mechanism avoids additional control exchanges and simplifies integration with TCP.

When data is being transmitted from the wireless host towards the base station, the absence of local acknowledgments means that losses can only be noticed when TCP duplicate acknowledgments arrive after an end-to-end round trip time, i.e. no earlier than in standard TCP. In addition, it is not clear any more whether the loss occurred on the wireless link due to wireless errors or elsewhere in the path due to congestion. A partial solution to this problem is for the base station to set an *explicit loss notification* (ELN) bit on duplicate acknowledgments corresponding to datagrams that never arrived via the wireless link and were presumably lost there. A modified TCP sender at the wireless host can in this case retransmit the lost segment without initiating congestion recovery [4]. Since only end-to-end acknowledgments are used however, in this direction recovery is much slower. If most data traffic is towards the wireless host however, recovery in the reverse direction is less critical.

One advantage of the Snoop protocol is that its understanding of TCP semantics, be-

sides economizing on bandwidth, eliminates the probability of competing retransmissions at both the local and end-to-end level [43]. Recovery is provided only for TCP segments that definitely require it, rather than every IP datagram. There are also many disadvantages to the Snoop approach, directly related to its reliance on violating layering by interpreting higher layer semantics, i.e. TCP sequence and acknowledgment numbers, at the link layer. Its slow recovery in the wireless host to base station direction is an even bigger issue for wireless links not at path endpoints or for multiple wireless link segments. Reliance on TCP acknowledgments also proves problematic when the TCP sender has no further data to send or it has exhausted its transmission window, since not enough duplicate acknowledgments are returned. The scheme is very tightly coupled with TCP mechanisms, and will require modifications as these mechanisms evolve. It is not applicable to any other protocol even if it has the exact same requirements as TCP since it relies on TCP mechanisms. Its performance over different types of wireless links is questionable since the same recovery mechanism is always used. It must separately keep track of each TCP connection passing through a base station since recovery depends on per connection sequence numbers. Finally, it cannot be used with encrypted IP packets that hide TCP headers [41].

Despite these drawbacks, the layering violation inherent in the Snoop protocol, labeled a *cross layer optimization*, is regarded by its proponents as a key technique in improving Internet performance over wireless links [44]. This is based on performance measurements showing that link layers unaware of TCP semantics are less efficient in terms of both throughput and overhead than Snoop [4]. Therefore, cross layer optimizations are *required* to optimize performance. We point out here the numerous problems arising due to this layering violation, which severely limits its applicability, in order to question the value of the Snoop approach at the design level. We come back to its performance in Chapter 4 where we show that awareness of TCP semantics is generally *not* required to optimize performance and that one way recovery is insufficient for the *most popular* TCP application on the Internet.

Cellular system RLPs avoid layering violations, and their close relation with the underlying physical link can be used to optimize their implementations. Local recovery is inherently faster than end-to-end recovery, especially with longer paths, since round trip delays over a single link are lower and more predictable, allowing tighter timeouts. Schemes providing full reliability however run the risk of retransmitting data so many times that the transport layer will retransmit it anyway, leading to duplicate retransmissions that waste bandwidth [43]. This can be avoided

using a limited retransmission approach as in the IS-95 RLP [5]. RLPs may waste bandwidth by retransmitting data for applications that either do not require additional reliability or prefer receiving new data segments rather than retransmissions of old ones. In general, with link layer error recovery schemes the hardest problem is to decide how much to enhance the underlying link. Adequate recovery must be supported to ease the task of reliable transports and to allow the realistic operation of unreliable transports, which assume only rare losses. Regardless of how fine tuned a link layer scheme is, it is probably impossible to design a *single* protocol that can cater to the needs of multiple transport layers and applications.

2.5 Requirements for a Universal Solution

Drawing upon the shortcomings of existing approaches to enhancing Internet performance over wireless links, we can formulate a set of general requirements that should be satisfied by any universal solution to these performance problems.

Location independence: A universal solution should not make any assumptions about the location of wireless links in an end-to-end path, the number of those links or the direction of data transfer. Such assumptions limit the applicability of the approach to very specific scenarios and can only provide partial performance improvements.

Adherence to layering: Layering violations restrict solutions to specific protocol semantics and mechanisms, preventing independent layer evolution. As encryption becomes more prevalent on the Internet, unrestricted access to higher layer information will gradually become impossible, rendering such approaches useless.

Easy deployment: Due to the size and heterogeneity of the Internet, new developments take years to propagate everywhere. A good solution should considerably improve performance without requiring large scale modifications to the Internet. It should be possible to initially deploy locally and spread incrementally.

Resource efficiency: A good solution should make efficient use of wireless link and host resources. It should minimize its state and processing requirements and avoid wasted overhead and retransmissions. The mechanisms used should be as close as possible to optimal for each particular wireless link.

Multiprotocol support: Not all transport layers have the same requirements. While TCP performance may be improved by repeated local retransmissions, real time UDP based applications may have stringent delay requirements coupled with relaxed reliability requirements, which are best served by limited recovery schemes.

Extensibility: The proliferation of wireless links will eventually cause many protocols to review their design assumptions and even new protocols to emerge. New applications may appear with unforeseen requirements. To serve such unanticipated needs, any solution should be easy to extend to cater to new needs.

Higher layer approaches fail to satisfy most of these requirements. They are protocol specific and too generic to be optimal for all links. Split connection schemes are also inefficient, due to their requirement for tracking per connection state, location dependent and unable to handle IP security. Transport layer specific link layer schemes such as Snoop are also limited solutions and inefficient due to their per connection state requirements. The Snoop protocol in particular is even more location dependent than split connection schemes and its layering violations make its future uncertain in view of IP security schemes.

Pure link layer schemes naturally satisfy most of these requirements. They are location independent since by definition they are only aware of a single link that may be located anywhere within an end-to-end path. They adhere to layering by not interpreting higher layer data. They are easy to deploy since they only require modifications at adjacent link endpoints. Link manufacturers or network operators can incorporate such schemes in their device drivers to gain a competitive edge for their systems. Detailed knowledge of the underlying link and its error and delay behavior enables the development of very efficient schemes.

While more promising than higher layer approaches, current link layer schemes are not multiprotocol solutions, and their extensibility is questionable. The main contribution of this dissertation is to show how these limitations can be overcome, thus satisfying all the above requirements. We show that link layer schemes are efficient solutions for numerous application and link type combinations (Chapters 4 and 5); that many such schemes can be combined in a *multi service link layer* to simultaneously serve diverse protocol needs in an extensible manner (Chapter 6); that the resulting solution remains efficient (Chapter 7); and that it is capable of supporting emerging and future higher layer Quality of Service requirements (Chapter 8).

Chapter 3

Experimental Setup and Methodology

This chapter describes the simulation environment and testing methodology used for the measurements presented in subsequent chapters. Section 3.1 presents the simulator we employed and its advantages and disadvantages compared to a real implementation. Section 3.2 describes the different types of links and error models we simulated. We discuss how our choices for model parameters were influenced by our goal to design a realistic and comprehensive test suite. Section 3.3 presents the network topologies we used in our tests and how each one examines a different aspect of the problems under consideration.

3.1 The Simulator

For the purposes of our research, we wanted to perform very extensive simulations of Internet applications over wireless links. When evaluating simulation platforms we were primarily looking for a system supporting reasonably accurate simulations of Internet protocols like TCP. Equally important was the ability to extend the simulator with arbitrary link layer modules and wireless link models to simulate various error recovery schemes and error generation processes. Secondary goals were simulator efficiency, especially with user extensions included, and support for automated testing of multiple scenarios. These features would allow us to perform as many different tests as possible within a limited time frame.

We decided to use *ns-2* [45], an event driven network simulator jointly developed by researchers at the University of California at Berkeley, the Laurence Berkeley National Laboratory and the University of Southern California. ns-2 is a freely available program running on

numerous operating systems and hardware platforms. For our simulations we used version 2.1b4 of ns-2, incorporating numerous additions that we developed, on a set of Pentium and Pentium II class workstations running the Linux OS with unmodified version 2.0.36 kernels. Since both ns-2 and Linux are available for free, other researchers can reproduce our tests on relatively inexpensive hardware. Note that our extensions to ns-2 are not Linux specific, hence they can be used with any platform supporting ns-2 without modifications.

ns-2 is particularly well suited to investigations of Internet protocol performance: its various TCP modules, although not reproducing every single protocol detail, quite accurately emulate the dynamic behavior of multiple current and proposed TCP variants [29]. ns-2 has been used to study TCP congestion control mechanisms [29], TCP performance over terrestrial wireless links [37, 39] and TCP extensions for satellite links [46]. To limit the scope of our research to manageable proportions, we decided to concentrate on wireless errors and their effects to higher layer performance. We avoided simulating congestion and dynamic routing, both important characteristics of the Internet in general and quite accurately modeled by ns-2. To reduce the significance of these omissions as far as possible, we did not modify TCP congestion control mechanisms in any way, and ensured that in all topologies simulated wireless links had to be part of the end-to-end path.

The most attractive characteristic of ns-2 for our purposes was that its source code is also available for free. ns-2 users can read the source code and assess whether the simulated objects are adequate for their purposes and, if needed, modify them to more accurately reflect their particular needs. It is also possible to add any functionality desired, such as link layer error recovery mechanisms, error models, link types, applications and statistics gathering modules. The simulator is written in C++ and Object Tcl, two object oriented languages, the former compiled and the latter interpreted. User visible objects, as opposed to internal simulator objects such as the simulation event scheduler, usually have both a C++ and an Object Tcl interface. To extend these objects, the user can use object inheritance to create new objects and add or modify object methods to implement any required behavior. Such extensions can be rapidly prototyped and debugged using interpreted Object Tcl, without recompiling and/or crashing the system. If additional speed is required, new objects can be ported to compiled C++ code for efficiency. We have used these features to their full extent, adding 5300 lines of C++ and 450 lines of Object Tcl code to ns-2 that extend existing and implement new objects. We describe our main addi-

```
stdout:1: ****************************
stdout:1: *** Simulation parameters ***
stdout:1: Topology: LAN1, Wireless link: Cellular
stdout:1: Application mix: SCFQ, Error level: 1
stdout:1: Primary LL: Default, Secondary LL: Default
stdout:1: *** Link options ************
stdout:1: LAN speed: 10Mb, LAN delay: 1ms
stdout:1: WAN speed: 2Mb, WAN delay: 50ms
stdout:1: Cellular speed: 14.4Kbps, Cellular delay: 100ms
stdout:1: PCS speed: 64Kbps, PCS delay: 50ms
stdout:1: WLAN speed: 2Mb, WLAN delay: 3ms
stdout:1: *** Simulation options ******
stdout:1: TCP type: TCP/Reno, TCP sink: TCPSink/MS, TCP tick: 0.5
stdout:1: Packet size: 50, FTP packets: 40000, Duration: 2000
stdout:1: PHTTP: 0, Max. connections: 4
stdout:1: CBR type: TwoState, CBR rate: 9.6Kbps
stdout:1: CBR on: 1s, CBR off: 1.35s
stdout:1: Error model: Uniform, Error rate: 0.01, Error unit: pkt
stdout:1: Error good duration: 10, Error bad duration: 0.1
stdout:1: BER good: 0.000001, BER bad: 0.01
stdout:1: Trace: 0, Nam trace: 0, Error trace: 0
stdout:1: TCP trace: 0
stdout:1: ****************************
```

Figure 3.1: Screen shot of simulator output

tions in the following sections and chapters, along with the simulations that employ them. Our extensions are freely available from the author as a *patch* file that transforms a stock ns-2 version 2.1b4 distribution to an exact replica of our own variant, with a single command [47].

Object Tcl, besides an object oriented prototyping tool, is also a scripting language with a full range of control structures, variables, output facilities, and so on. Since it has direct access to simulated objects, it is an ideal tool for setting up and controlling simulations. The ns-2 executable contains an embedded Object Tcl interpreter that can execute scripts in interactive or batch mode. A simulation script creates objects such as nodes, links, protocols, applications and error models, sets their parameters, starts the simulation, and gathers statistics during and after execution, all using Object Tcl commands. The power of this approach allowed us to write a single, but quite large, parameterized Object Tcl script and use it to execute *all* of our tests, which amount to tens of thousands of test cases. Figure 3.1 shows part of the output from one such test case. A second, very short, script with a few nested loops can generate all these test case parameters and pass them to the main script for execution. Hence, with a single command, and lots of patience, other researchers can fully reproduce our measurements.

3.1.1 Simulation vs. Implementation

Any network simulation study automatically raises questions about the accuracy of its models, and consequently the validity of its results. For efficiency reasons, not all aspects of the system can be simulated exactly: some features are omitted, while others are modeled only approximately. Although the limitations of ns-2 and its built in modules are well known, it is accepted as a valuable experimentation tool by the Internet research community, as evidenced by the numerous published performance studies that use it, its continuing evolution and its active user community. For our part, we avoided modifying existing ns-2 behavior and attempted to be as close to reality as possible in our own extensions. We already mentioned that we ignored congestion and dynamic routing to simplify our test suite. We identify further such limitations as we describe the relevant simulation objects. We believe that despite the inherent limitations of the simulation approach, our results reflect models accurate enough to be valid and are therefore valuable to the Internet research community.

The main reason for our choice of simulation over writing and testing a real implementation of our proposals was that we wanted to avoid the limitations of previous Internet performance over wireless studies by being *very* comprehensive in the range of tests performed. We tested multiple link layer schemes with a variety of links, error models, error levels, transport protocols, applications and network topologies. Previous studies only tested a single application and transport protocol over one type of link and error model, with a few error rates and network topologies. The applicability of such results is necessarily limited to a specific test setup. Our results indeed reveal that there are no universal solutions: the best solution in each case depends on multiple test parameters, some of which had been completely overlooked in less comprehensive studies. The volume of our undertaking prohibited writing and testing real implementations in terms of both development and test execution time. In contrast, with the ns-2 simulator we were able to prototype numerous link layer schemes and even a completely novel link layer architecture, and test them under hundreds of different scenarios, using automated scripts.

Simulation also has some definite advantages over testing of real implementations. One such advantage is that wireless links that are not available to the experimenter can also be tested. This allowed us to simulate wireless links that were either not present in our service area or were still in development. Similarly, we were able to perform tests under various theoretical error models using a wide range of error model parameters. Theoretical error models are only

an approximation of reality, but since their behavior is based on a mathematical random number generator, it can be exactly reproduced numerous times by using the same seed for the generator. This allows different error recovery schemes to be tested under the exact same sequence of errors, easing comparisons between results. Conversely, to avoid drawing conclusions from results dependent on a single error sequence, tests can be repeated with different random number generator seeds and their results statistically analyzed.

In our work, we repeated each and every individual test scenario 30 times, using a different random number generator seed for each run. We then calculated the mean and other statistics (such as variance and standard deviation) of the performance metrics under measurement. We always used the first 30 "good" random number generator seeds included in ns-2. These seeds are guaranteed to provide statistically independent pseudo random sequences. To isolate the behavior of the error models from that of other simulation objects that used statistical models, for example traffic generators, we used separate random number generators for the latter. These generators were always seeded with the same number for all repetitions of a test scenario, so that the behavior of those objects would remain as far as possible the same, despite changes in error model behavior. We point out the objects controlled in this manner in the relevant simulation sections.

3.2 Links and Error Models

Previous studies of Internet performance over wireless concentrated on a single type of wireless link and error model. To answer the question of whether one error recovery mechanism is optimal for all types of links or not, we decided to simulate multiple wireless links with widely varying bandwidth, delay and error characteristics. The links and error models chosen reflect a compromise between accurate renditions of real links, resemblance to models used in previous studies and implementation simplicity. We have simulated a slow digital cellular link, a fast wireless LAN and a projection for a next generation personal communications system link. The error models used are uniform frame losses, exponential intervals between bytes in error and two state (good/bad) uniform bit error rates with exponential state durations, respectively. Each link was tested with four different sets of error model parameters. In the following paragraphs we describe each link and error model employed, their parameters, and explain our choices.

Transmission Speed	14.4 Kbps
Propagation Delay	100 ms
Frame Size	50 bytes
Error Model	Independent Frame Losses
Loss Rates	
Error Level 1	1%
Error Level 2	2%
Error Level 3	5%
Error Level 4	10%

Table 3.1: Cellular link characteristics

3.2.1 Digital Cellular

The digital cellular telephony link simulated, referred to in the simulation results simply as "Cellular," reflects a mix of characteristics from various existing systems. We set propagation delay to 100 ms, a value quoted for one way delay in the IS-95 system [13]. This relatively high delay is not due to signal propagation but due to the time consuming process of bit interleaving and de-interleaving among multiple link frames to make embedded FEC schemes effective. We set the transmission speed of the link to 14.4 Kbps in full duplex transparent mode, slightly higher than most existing cellular systems, to allow voice (compressed using the very efficient IS-95 voice encoder) and data to co-exist on the same link. This value is very close to the 13 Kbps available to voice channels in GSM [21], as opposed to the 9.6 Kbps available for the non-transparent data service. Since we assumed use of a transparent mode, we had to keep frames small, but somehow adequate for higher layers. We settled on a 50 byte maximum frame size, excluding any link layer overhead.

The error model used on this link is similarly borrowed from IS-95: frames are lost based on an independent error model with loss rates set at 1%, 2%, 5% and 10%. The 1% and 2% error rates are typical for IS-95 with 20 byte frames. Extrapolating these rates to 50 byte frames we arrive at the 5% rate. The 10% rate is added as an extreme case to study how the schemes perform under very harsh conditions. The uniform frame error loss assumption is justified by the bit interleaving and FEC techniques used at the physical layer. This error model was simulated using an existing ns-2 module: every time a frame is received, a sample is drawn from a uniform distribution to make a choice between accepting and dropping the frame. Since the model is

Transmission Speed	64 Kbps
Propagation Delay	50 ms
Frame Size	250 bytes
Error Model	Two State with Exponential Durations
Bit Error Rate in Good State	10^{-6}
Bit Error Rate in Bad State	10^{-2}
Average Good State Duration	10 seconds
Average Bad State Durations	
Error Level 1	100 ms
Error Level 2	200 ms
Error Level 3	500 ms
Error Level 4	1000 ms

Table 3.2: PCS link characteristics

stateless, it is reasonable to only trigger it on frame arrivals. The characteristics of Cellular links are summarized on Table 3.1. We generally refer to the different loss scenarios for each error model as *error levels* in the simulations, with error level 1 reflecting the best link conditions and level 4 the worst.

3.2.2 Personal Communications System

The personal communications system cellular link simulated, referred to in the simulation results as "PCS", is a projection of what a next generation PCS link could look like. The bandwidth was set to 64 Kbps full duplex, based on the capability of GPRS [19] and Wideband CDMA [20] systems to allocate multiple channels to a single user. This bandwidth could be viewed as, for example, the result of aggregating 8 channels with 8 Kbps of bandwidth each, after excluding synchronization and multi link aggregation overhead. This is sufficient to simultaneously support data and voice, even using very simple voice encoders. We arbitrarily set the propagation delay to 50 ms, half that of Cellular, to reflect the more friendly approach to Internet protocols adopted by next generation cellular systems. Due to the increased speed of the link compared to Cellular, we also increased the maximum frame size for PCS to 250 bytes excluding any link layer overhead, always assuming a transparent mode service.

For errors, we wanted to use a model reflecting fading, much like a raw link would without bit interleaving and FEC. We decided on a two state model: in the "good" state, bit errors occur based on an independent model at a 10^{-6} rate, while in the "bad" state, the bit

error rate increases to 10^{-2}. As a result, during bad periods frames are lost with near certainty, while during good periods the frame loss rate is only 0.2%. The duration of both states is calculated from an exponential distribution: the average good state duration is 10 seconds, while the average bad state durations are 100 ms, 200 ms, 500 ms and 1000ms. These translate to bad states lasting on average for 800-8000 bytes. Although the ratio between good and bad state duration seems to be close to the error rates for Cellular, this error model is significantly harsher, as the channel can become completely unusable for the duration of multiple frames. This allows us to study the effects of clustered errors on both link layer schemes and higher layer protocols and applications. Similar two state models with exponential state durations and the same good and bad state bit error rates have been used by other researchers in performance studies [37, 39]. The characteristics of our PCS links are summarized on Table 3.2.

To simulate the two state error model we had to extend the two state error model of ns-2 which assumed that during the bad state all bits were corrupted, while in the good state no bits were corrupted. Thus any frame coinciding with a bad state had to be dropped. We instead calculated an effective error rate for each frame based on the bit error rate of each state, and then drew a sample from a uniform distribution to decide whether to accept or drop the frame. Since a frame could span multiple good and bad states, we had to advance the state of the error model enough to fully cover frame transmission, identify the bit lengths subject to each state, calculate an error probability for each, and then combine them all before making a final decision. To further enhance the accuracy of the model, instead of advancing the model state based on the number of bits received, which would "freeze" the state during periods of inactivity, we advanced the state based on real time. For example, consider a frame transmitted at the beginning of a bad period lasting for two frame times. With a bit based model, if the link is idle until the frame is retransmitted, the retransmission will also be lost regardless of when it occurs. With a time based model, the retransmission's fate will be decided by the actual state of the model at retransmission time.

3.2.3 Wireless Local Area Network

The last simulated link is a Wireless LAN, referred to in the simulation results as "WLAN." Although we borrowed many parameters from the WaveLAN [6], our WLAN link is different: we assumed a full duplex point-to-point link instead of a shared multi-point link.

Transmission Speed	2 Mbps
Propagation Delay	3 ms
Frame Size	1000 bytes
Error Model	Exponential Intervals between Byte Errors
Average Intervals between Byte Errors	
Error Level 1	131072 bytes
Error Level 2	65536 bytes
Error Level 3	32768 bytes
Error Level 4	16384 bytes

Table 3.3: WLAN link characteristics

The point-to-point assumption is not a problem for our purposes, since we only needed to model two hosts. The full duplex assumption however changes the characteristics of the link, since contention for the link between forward and reverse traffic is eliminated. Our past research has uncovered serious problems with contention in the WaveLAN [11], but unfortunately such problems are *not* modeled by ns-2. Furthermore, we found the LAN implementations of ns-2 to be quite unreliable. For example, the Snoop module included in the ns-2 distribution, which was explicitly written for LANs, would crash ns-2 whenever used. Thus, we settled for a duplex link with WLAN parameters, with the explicit understanding that our study ignores link contention, collisions and other MAC layer effects. The WLAN link bandwidth was set to 2 Mbps in full duplex mode, and the propagation delay to 3ms. We set the maximum frame size to 1000 bytes excluding any link layer overhead.

We wanted an error model reflecting the very good error behavior of the WLAN [10, 11]. We chose to simulate single byte errors occurring at exponential intervals, to facilitate comparisons with previous WLAN performance studies [4]. Drawing upon the same studies we set the average durations between errors to 131072 (2^{17}) bytes, 65536 (2^{16}) bytes, 32768 (2^{15}) bytes and 16384 (2^{14}) bytes. Note that the smaller the interval, the harsher the error conditions. For 1000 byte frames, these roughly translate to frame error rates of 0.7% to 6.1%. We implemented an appropriate error model for ns-2, but in contrast to the PCS case, we made the WLAN model byte rather than time based, so that its state only advances when frames are transmitted. Again, this choice was made so as to enable comparisons with previous studies. This limitation is not serious considering the very low error rates simulated and the instantaneous (one byte) duration of the errors. The characteristics of WLAN links are summarized on Table 3.3.

3.2.4 Link and Error Model Limitations

Besides link specific limitations noted in previous paragraphs, all link and error models share some limitations. For simplicity, we decided to use a maximum frame size for each link, and forced higher layer protocols and applications to generate packets of this size to avoid simulating IP fragments, a feature unsupported in ns-2 and considered inefficient on the Internet [48]. We also ignored IP, UDP and TCP headers in all cases, assuming that each datagram consists only of user data, a common assumption in studies using ns-2. Although this influences the actual results obtained, since all link layer schemes are subject to it their relative performance remains the same. A more questionable issue is whether it is reasonable to ignore TCP and IP headers amounting to at least 40 bytes when using 50 byte Cellular frames. If however header compression [31] is employed over these links, these headers are reduced to 3-5 bytes, hence the overhead per frame remains reasonable.

A related limitation is that we ignored any link layer framing overhead except headers explicitly added by the simulated schemes. Such overheads include framing flags, CRCs and so on. Since these are fixed regardless of the link layer scheme used, they do not significantly influence test results. Headers added by the simulated schemes are explicitly taken into account for both transmission time and error probability calculations, but they are (unrealistically) added to the maximum frame sizes. Since however all protocols simulated use very small frame headers (1-3 bytes), this is not a significant problem.

A more important limitation is imposed by the nature of links in ns-2. Full duplex links are simulated as two independent simplex links, and this extends to their error models. As a result, losses do not occur at the same time in both directions. For example, in PCS links the bad states simulating fade periods are independent in each direction: they start at different times and last for different periods. Although it may be argued that fading periods in duplex links that employ frequency division duplexing (i.e. the uplink and downlink use different frequencies) are not completely synchronized, symmetric error models would probably be much more realistic.

3.3 Simulated Network Topologies

Previous performance studies have been limited in the types of network topologies studied. Usually a single topology consisting of one wireless link situated at the edge of a wired

local area or wide area path was tested. We decided to test both LAN and WAN based wired paths, as previous research has shown that the higher end-to-end delay of WAN paths has a significant effect on transport layer performance [4]. In addition, we tested those paths with either a single wireless link at one end or two wireless links, one at each end. The latter configuration, besides pointing out the performance degradation due to the inclusion of multiple error prone links, can also reveal some performance shortcomings of location dependent schemes. In the following paragraphs we describe the topologies used, their parameters, and explain our choices.

3.3.1 One Wireless Link Topologies

The simplest network topology to simulate, used in virtually all previous performance studies, consists of a single wireless link at the edge of an end-to-end path. The rest of the path is either a single LAN link or a longer WAN path. Since we decided to avoid simulating congestion effects in our tests, we abstracted the multiple hops of the WAN path into a single link with lower bandwidth (due to multiplexing and long haul link limitations) and higher delay (due to multiple hops) than the LAN link. This topology is depicted in Figure 3.2: two hosts communicate via either TCP or UDP, with the wireless host being attached to a base station that acts as a regular IP router. The wireless link may be any of the three types of link presented above. The wired link is either a 10 Mbps LAN link with 1 ms propagation delay, a topology referred to as LAN1 in the tests, or an (abstract) 2 Mbps WAN link with a 50 ms propagation delay, a topology referred to as WAN1 in the tests. The LAN link has the characteristics of a conventional Ethernet. The speed of the WAN link was chosen to match the fastest wireless link (the WLAN) to avoid making it a bottleneck, while the delay was set based on informal measurements of the round trip time between hosts separated by a transcontinental path.

From previous studies we know that TCP performance is further degraded by wireless errors with longer end-to-end paths [4]. This is inherent in end-to-end error recovery schemes: the larger the round trip delay, the longer it takes for TCP to recover from a loss, given its drastic reaction to losses described in Section 2.2. By using both the LAN1 and WAN1 topologies, we can evaluate to what extent link layer schemes can reduce this disadvantage of longer paths. In both topologies, we have chosen to set the main traffic direction from the wired towards the wireless host. For unidirectional UDP or TCP data traffic this means that the wired host is the

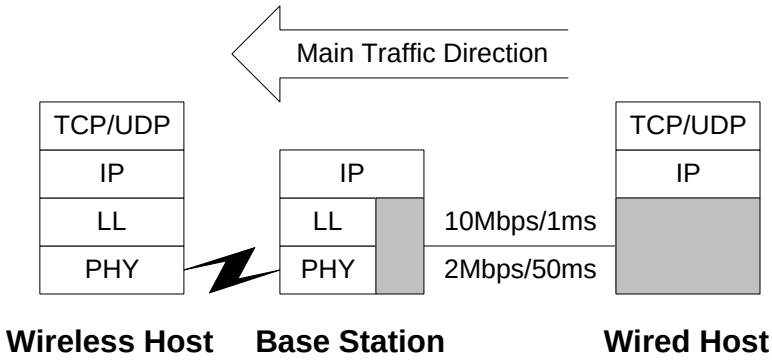

Figure 3.2: One wireless link topologies (LAN1 and WAN1)

data sender and the wireless host the data receiver. For bidirectional applications, this means that most data traffic flows from the wired towards the wireless host. Although we have pointed out in Section 2.5 that some schemes by being location dependent are only profitable for this traffic direction, we do not simulate the reverse scenario since it is a special case of the two wireless link topologies.

3.3.2 Two Wireless Link Topologies

The two wireless link topologies are simple variations of the one link topologies discussed above, as depicted in Figure 3.3. Both ends of the end-to-end path are wireless links, with the remainder of the path consisting of either a single LAN link, a topology referred to as LAN2 in the tests, or an (abstract) WAN link, a topology referred to as WAN2 in the tests. The wired links are characterized by the same parameters as in LAN1 and WAN1, that is 10 Mbps bandwidth and 1 ms delay or 2 Mbps bandwidth and 50 ms delay, respectively. The wireless links can be of any of the three types presented above, but we decided to limit our tests to topologies where *both* links are of the same type. The reasoning was that with two different wireless links the least capable of the two would become the bottleneck, effectively determining end-to-end performance. Hence, performance with these topologies would lie between those of the one and two wireless link topologies using the least capable link only. In order to be able to separately evaluate each link layer scheme in two wireless link topologies, we also decided to use the same scheme for both links in any given test. The error models of the two wireless links are indepen-

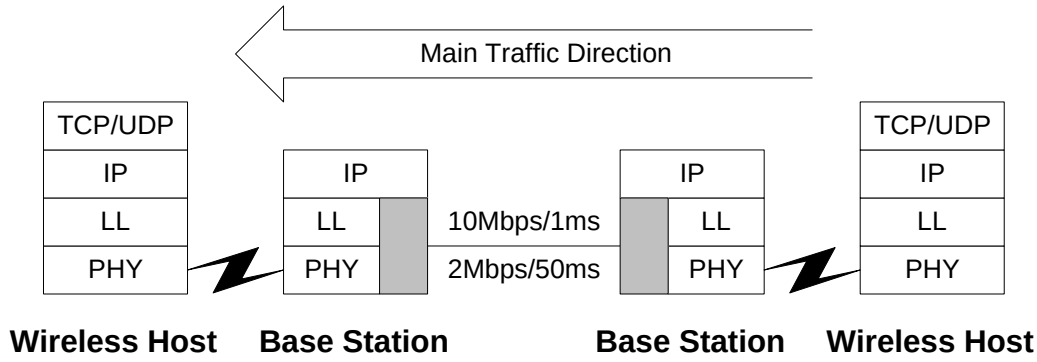

Figure 3.3: Two wireless link topologies (LAN2 and WAN2)

dent. Although we set the main data traffic direction to always be from the same simulated node to the other, these topologies are completely symmetric so the traffic direction does not really matter.

Besides depicting how errors with two wireless links further degrade end-to-end performance, symmetric topologies reveal any location dependencies inherent in error recovery schemes. With wireless links at both ends of the path, all data have to cross them in both the base station to wireless host direction and its reverse. As a result, schemes that only provide base station initiated error recovery show their limitations with these topologies regardless of the dominant direction of data transfer [37, 39, 42]. The two link topologies are generalizations of the one link topologies presented above, both with and without reversing the main traffic direction. This is why to reduce the number of test cases we only tested a single traffic direction over one link topologies. Similar observations could be made by using topologies with *transit* wireless links, i.e. end-to-end paths that include an intermediate wireless link. None of the endpoints of a transit link coincides with an end host, and there is no inherent asymmetry with respect to the volume of traffic flowing in each direction. We felt however that the topologies adopted were more realistic for the end user oriented links that we are studying, expecting that high speed satellite links would make more sense for transit topologies.

Chapter 4

TCP Application Performance

This chapter describes the TCP applications and link layer schemes we simulated, presents detailed simulation results and analyzes their implications. Section 4.1 discusses the applications used and their simulator implementations. Section 4.2 describes the link layer schemes implemented and tested. Section 4.3 presents and analyzes our simulation results. Section 4.4 summarizes our conclusions and contrasts them with those of other researchers.

4.1 Simulated TCP Applications

As discussed in Section 2.2, TCP offers a reliable byte stream service, resembling reading (writing) from (to) a sequential file. This interface tremendously eases the task of programming network applications as it effectively masks IP limitations. As a result, numerous applications employ TCP as their transport protocol, for example file transfer, e-mail, remote terminal emulation and World Wide Web browsing. Since TCP performs error, flow and congestion control transparently to higher layers, to a large extent the behavior and performance of these applications is determined by TCP itself. Applications with requirements that cannot be satisfied by TCP, such as audio/video conferencing with their stringent delay limits, usually employ UDP and add their own mechanisms for error, flow and congestion control. We discuss these applications and their performance in Chapter 5.

Previous performance studies have, explicitly or implicitly, made two key assumptions on the behavior of TCP applications. First, since TCP is in complete control of all network related issues, improving TCP performance under a single application will have similar benefits

for all other TCP applications. Second, wireless hosts at the periphery of the network are clients of servers residing in the wired part of the Internet, therefore data is mostly transferred in the wired to wireless direction. Combining these two assumptions, researchers chose bulk data transfer from a wired server to a wireless client as a representative application to test and attempt to improve. We felt that this scenario was not adequate to predict the performance of *interactive* applications, where data transfer in *both* directions is equally important, even if the transfer sizes are asymmetric. For this reason, we decided to simulate both a bulk data transfer and an interactive application. These applications are described in the following paragraphs.

4.1.1 File Transfer

Files can be transferred through the Internet by using the *File Transfer Protocol* (FTP) [49]. Users execute a file transfer application that uses FTP for the actual transfers. FTP in turn uses TCP to copy the files between the hosts. A file transfer application offers a variety of interactive features, such as listing files, changing directories, and of course sending and receiving files. However, it is the performance of individual file transfers that determines application performance as a whole. Hence, in common with all previous performance studies, we only simulated the file transfer phase of FTP. Essentially this means the transmission of a specified amount of data from a sending host to a receiving host. We used the existing FTP object in ns-2 to model these transfers. The implementation of this object is illuminating: whenever TCP can send more data, FTP provides them immediately, until the transfer completes. As a result, FTP behavior and performance is fully determined by TCP, explaining its widespread use in studies of TCP performance. For one wireless link tests (LAN1 and WAN1) we set the file transfer direction to be from the wired towards the wireless host. The reverse direction is included as a special case in the two wireless link tests (LAN2 and WAN2) where the file is transferred between the two wireless hosts.

The main performance metric for file transfers is *throughput*, defined as the amount of data transferred by the *application* divided by the time interval between sending the first data packet and receiving the last acknowledgment packet. Note that TCP may actually transmit an amount of data that *exceeds* the amount of application data transferred. This includes retransmissions of lost data and duplicate copies of data that were considered lost by TCP. However, the user is only interested in the amount of application data transferred, which is reflected in

Link Type	Transfer Size	Packet Size	Number of Packets
Cellular	2 Mbytes	50 bytes	40,000
PCS	10 Mbytes	250 bytes	40,000
WLAN	100 Mbytes	1000 bytes	100,000

Table 4.1: File transfer parameters

out throughput definition. For fixed size transfers, throughput and time taken are equivalent, but throughput is preferred as a metric because it can be compared to nominal link bandwidth. Due to TCP's dramatic reaction after losses, instantaneous FTP throughput continuously fluctuates in high loss environments, hence long transfers are needed in order to get good estimates of average throughput. In practice users will not initiate file transfers that would take days to complete. We thus decided to use different file sizes in our FTP tests for each type of wireless link. The file sizes chosen for each link are shown in Table 4.1, along with the number of packets they represent and their size. Note that, as mentioned in Section 3.2, we forced TCP and FTP to use the same packet size as underlying link frames. The minimum time for these transfers, ignoring losses and TCP dynamics, is 18.5 minutes for Cellular, 20.8 minutes for PCS and 6.7 minutes for WLAN, striking a balance between acceptable transfer times and reasonably accurate throughput results.

Link layer schemes also introduce overhead due to retransmissions, link layer headers and (possibly) control frames. These data are also not included in application throughput calculations. However, due to the limited bandwidth of wireless links, it is important to minimize such overhead. For this reason we measure the *goodput* of file transfer tests, defined as the (fixed) amount of application data transferred divided by the total amount of data sent over the wireless link, including TCP and link layer retransmissions and overhead. We used the same definition of goodput as some previous studies to allow comparisons [4]. Goodput is a fraction that diverges from its optimal value of 1 by at least the frame error rate, since we need to retransmit all losses.

4.1.2 World Wide Web Browsing

An important factor differentiating our study from previous ones is that we decided to simulate an interactive application. This was expected to clarify whether unidirectional file transfer is an adequate model of TCP application performance. We chose World Wide Web

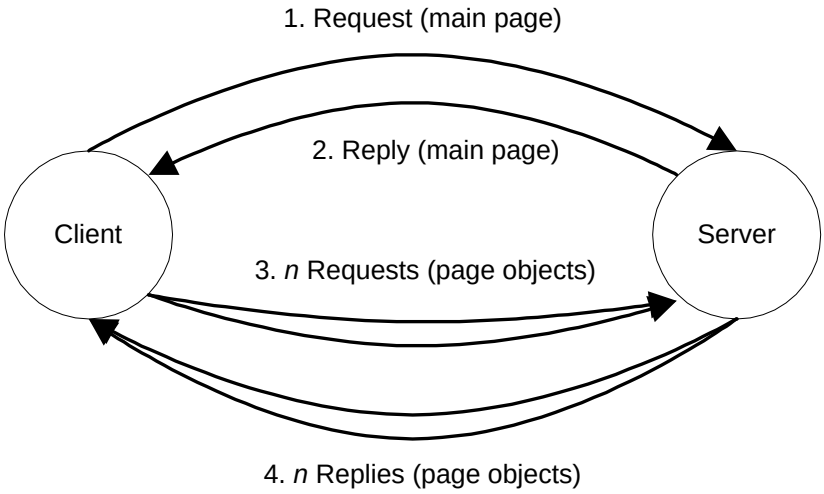

Figure 4.1: Sequence of events in an HTTP transaction

(WWW) browsing for this purpose. According to 1995 traffic statistics of the NSFnet backbone, quoted in [50], this was the *most popular* application in terms of both IP datagrams and total amount of data transferred, and its popularity seems to be increasing on a daily basis. In WWW browsing, a user employs a client application to look at *pages* stored at a WWW server. Pages consist of text, embedded objects and *links* to other pages. Selecting a link causes the target page to be displayed. Clients and servers use the *HyperText Transfer Protocol* (HTTP) [51] to communicate. HTTP defines request and reply messages used to ask for and deliver pages and objects embedded in them, such as images, audio files or scripts. HTTP in turn uses TCP connections to exchange messages and files. We can model a WWW browsing session as a sequence of *transactions*: the user selects a link, causing a message to be sent to the server requesting a page; the server responds with the page; the client requests each one of the n embedded object separately; the n embedded objects are returned, completing the transaction [50]. This sequence of events is depicted in Figure 4.1. The user may pause for some time and then start a new transaction by selecting another link.

Although this process can be modeled as a sequence of small file transfers, it is profoundly different to FTP: the sequence of events is important, meaning that the request has to be received by the server before a page and its elements can be returned. The small file sizes also mean that the throughput of individual transfers may not reach the peak achievable value due

to TCP start up dynamics. Therefore, WWW browsing performance is influenced by both TCP behavior and user actions. Furthermore, even though most data flows from the server towards the client, the client's requests are equally important to the transaction. Hence, despite the asymmetric data loads, WWW browsing is an inherently bidirectional application, so in order to improve its performance over wireless links *both* directions must be considered. Similar observations hold for other interactive applications such as Telnet [52], a remote virtual terminal application, where user input must be received before the server can return screen updates. Interestingly, non interactive applications that do not involve user input, such as downloading e-mail using the *Post Office Protocol* (POP) [53], are also dependent on link performance in both directions: the e-mail client must request each message separately before the server can return it. Therefore, relying on FTP tests only is a serious limitation of a performance study.

For our tests we used the empirical HTTP traffic distributions described in [50]. These distributions can be used to generate request sizes, reply sizes (including both pages and embedded objects) and the number of embedded objects per page. A module supplied by ns-2 encapsulates these distributions into a session between a client and a server, automatically generating proper sequences of requests and replies and using TCP to transfer the data. The module can either use individual TCP connections for each HTTP transfer or reuse a *persistent* connection for the duration of a transaction. Similar to file transfer, we set both HTTP and TCP packet sizes equal to the frame sizes for each wireless link (see Table 4.1). To increase the number of modeled transactions we set the user think time to zero, meaning that after a transaction completes a new one is initiated immediately. In order to draw samples from the HTTP distributions, a random number generator is used. We created a separate generator which was always seeded with the same value, to isolate the HTTP sequence of events from changes in the seed for the error models. This does not guarantee the exact same sequence of events however: the order in which the generator is consulted to draw embedded objects sizes is determined by the arrival of requests which in turn depends on the error models. For one wireless link tests (LAN1 and WAN1) we set the client at the wireless host and the server at the wired host, so that most, but not all, traffic would flow towards the wireless host. The reverse direction is included as a special case in the two wireless link tests (LAN2 and WAN2) where both server and client are located on wireless hosts.

Each HTTP transaction consists of a statistically determined number of transfers and

transfer sizes. Since we cannot guarantee that the sequence of HTTP events will always be the same regardless of changes in error behavior, counting the number of transactions over a period of time does not produce a useful metric. We decided instead to measure HTTP *throughput*, defined as the total amount of HTTP data transferred from the server to the client divided by the time taken for these transfers. Note that, as in file transfers, throughput includes only application data, not TCP and link layer overhead. In the reverse direction only request messages are transferred, which were ignored. For each type of wireless link we simulated the system for a fixed period of time before calculating this metric, starting the first transaction at time zero. These periods were 2000 seconds for Cellular and PCS and 500 seconds for WLAN. Since the ns-2 module only reports the amount of data transferred at the conclusion of each transaction, at simulation end there could be a single transaction in progress at an unknown stage. Note that a new transaction is only initiated after the previous transaction has completed, hence one and only one transaction is in progress at any given time. To avoid skewing our results, we ignored the interval after the last complete transaction ended, hence the throughput metric reflects total data transferred in *completed* transactions divided by the time taken by these transactions. We did not measure goodput for HTTP since due to the bidirectional nature of the transfers, data and acknowledgments are mixed in both directions in an unpredictable manner.

4.2 Simulated Link Layer Schemes

In order to choose link layer schemes suitable for TCP applications, we mainly need to consider the requirements that TCP itself imposes. TCP offers a fully reliable service, so link reliability must be maximized to avoid its adverse reaction to losses. However, since TCP may timeout anyway after long delays, it may be preferable to eventually abandon recovery at the link layer under persistent error conditions so as to avoid competing retransmissions [43]. TCP receivers generate duplicate acknowledgments on reception of out of sequence data, which are used for loss detection at the sender, so it is preferable to deliver data in sequence. In the following paragraphs we describe the link layer schemes that we chose to implement and test. These include conventional full recovery schemes, a more recent limited recovery scheme, and the Snoop scheme which, unlike the rest, exploits knowledge of TCP semantics. We also discuss other schemes that we either did not implement or test. All simulated schemes include at least 1

byte of overhead in each frame, containing a link layer scheme identifier which takes up 4 bits, leaving the rest available. The use of this identifier is discussed in Chapter 6. Since this overhead applies to all schemes, it does not influence their relative performance results.

4.2.1 Full Recovery Schemes

Traditional *automatic repeat request* (ARQ) link layer schemes offer full recovery, i.e. they keep retransmitting lost frames indefinitely. They are usually based on a *sliding window* principle: incoming frames are numbered and transmitted in order of arrival, with a copy placed in a fixed size buffer array. The size of this array, or *window*, determines how many frames may be outstanding at the sender. When the earliest frame of the window is acknowledged, its buffer is freed and the window advances, or *slides*, to accept a new incoming frame. This implies that the size of the window must be large enough to allow continuous frame transmission until this acknowledgment arrives, else the scheme will exhaust its window and stall, leaving the link unused. In the simplest variant of this scheme, the receiver accepts frames strictly in sequence. The sender sets a retransmission timer after each transmission, equal to at least the round trip delay of the link. If this timer expires before an acknowledgment is received, the sender assumes that the frame was lost, and retransmits it. Already transmitted frames with higher sequence numbers must have been dropped by the receiver, so they must also be retransmitted. As a result, each timeout causes the scheme to restart from the earliest buffered frame, hence the scheme is called *Go Back N* (GBN) [15], where *N* stands for the buffer size.

Our full duplex GBN implementation adds two extra bytes of overhead to each data frame, one for the sequence number and one for a piggybacked acknowledgment. The receiver also maintains a delayed acknowledgment timer which is set when a new frame arrives. If a data frame with a piggybacked acknowledgment has not been sent before timer expiration, a pure acknowledgment frame is sent, containing only the two header fields. The frame type (data or acknowledgment) is stored in the 4 unused bits of the overhead byte that also contains the scheme identifier. GBN supports variable size frames, although data frames always use the maximum size. The main drawback of GBN is that it wastes bandwidth by retransmitting all outstanding frames after each loss. Since the retransmission timer is at least equal to the round trip time, all frames transmitted during that interval are wasted. To avoid this limitation, we can add a buffer at the receiving side to store frames received out of sequence. Since we want to deliver frames

in sequence to higher layers, this buffer also works as a sliding window: the earliest frame in sequence must be received before advancing the window. The receiver buffer allows the sender to avoid retransmitting frames that have already been received. After the lost frame arrives, the receiver advances the window by releasing to higher layers all frames received in sequence and sending an updated acknowledgment to the sender. This scheme is called *Selective Repeat* (SR) [54] since it allows specified frames to be retransmitted.

Our implementation of SR also extends GBN by using *negative* acknowledgments. Instead of waiting for the sender to timeout and retransmit an unacknowledged frame, when the receiver receives an out of sequence frame, it sends a negative acknowledgment to the sender with the sequence number of the earliest missing frame. Unlike regular acknowledgments, these frames are never piggybacked on data frames, but like pure acknowledgments, they only contain two header bytes. Since only the earliest frame may be negatively acknowledged, this implies that all previous frames have been received. The sender can retransmit negatively acknowledged frames immediately, without relying exclusively on timeouts, although timeouts are still required for the cases where negative acknowledgments themselves are lost. To avoid asking the sender multiple times for the same frame before it has time to respond, after a negative acknowledgment is sent a new one cannot be sent until the receiver's window advances, hence only one negative acknowledgment may be sent for each frame. This scheme is much more efficient than GBN due to its avoidance of wasted retransmissions and faster loss recovery.

A limitation of this basic SR scheme is that when multiple frames are lost in a single window, only the first one is negatively acknowledged. The receiver's window must advance before sending a new negative acknowledgment, hence only a single loss can be recovered from in one round trip time. To speed up recovery in these cases, we implemented a variation of SR that allows multiple negative acknowledgments per window: every time a "hole" appears in the sequence numbers of received frames, negative acknowledgments are sent for all missing frames, in strictly ascending sequence [55]. The ascending sequence guarantees that only a single negative acknowledgment may be sent for each lost frame. Since in this scheme negative acknowledgments do not refer to the earliest frame expected, they do not imply that all previous frames are received, hence separate acknowledgments are always needed.

Even with multiple negative acknowledgments per window, when negative acknowledgments or retransmissions are lost, this scheme has to rely on the inherently slower timeout

initiated recovery. A further improvement on the SR theme is to allow multiple negative acknowledgments per lost frame. The key observation is that since negative acknowledgments are sent in ascending order, the corresponding retransmissions should also arrive in the same order. If a missing frame arrives and it is not the earliest frame expected, we can conclude that earlier missing frames were either not retransmitted (due to a lost negative acknowledgment) or their retransmissions were also lost. Hence, they should be negatively acknowledged again. To make this scheme work despite the fact that negative acknowledgments are no longer sent in strictly ascending sequence, we must keep track of the number of times each lost frame was negatively acknowledged. We only send new negative acknowledgments for frames that have been negatively acknowledged fewer or the same number of times as the newly received frame, and increase their counters [55]. Both these variations on basic SR are quite simple to implement despite their apparent complexity and they are potentially very useful for harsher error environments. They use the same frame formats and headers as SR. We refer to the first scheme (multiple negative acknowledgments per window) as SR/M1 and to the second scheme (multiple negative acknowledgments per frame) as SR/M2.

4.2.2 Limited Recovery Schemes

The conventional ARQ schemes presented above improve TCP performance because the round trip time of a single link is much smaller and more predictable than that of an end-to-end path, allowing them to retransmit the same frame multiple times before TCP times out. Under persistent error conditions however, TCP may timeout anyway and retransmit the same data that the link layer is retransmitting, leading to competing retransmissions which waste bandwidth [43]. One way to avoid this problem is to rely on knowledge of TCP semantics to identify TCP retransmissions, and avoid retransmitting the same data twice. This is the approach taken by the Snoop scheme [42]. Another option is to give up on frames that have not been received after a number of retransmissions, letting higher layers like TCP deal with the loss. This is the approach taken by the IS-95 RLP [5]. Since a pure link layer scheme is unaware of TCP semantics, it has to choose a specific maximum limit for retransmissions that will be safe even for short end-to-end paths, while Snoop can keep trying until a TCP retransmission actually arrives. On the other hand, a pure link layer scheme may be used without modifications to support any reliable higher layer protocol. In addition, it can be used for applications that expect only limited

reliability from the network.

Our implementation of the IS-95 RLP, referred to simply as RLP, follows its published descriptions [5, 13, 56]. The sender has a retransmission buffer, but it does not maintain retransmission timers and has no notion of a window: new frames are copied in a circular buffer and get the next number in sequence. Buffered frames are retransmitted when they are negatively acknowledged. The receiver maintains a window, but it only sends negative acknowledgments when "holes" appear in this sequence, in the same manner as SR/M1, not regular acknowledgments. This allows the elimination of the piggybacked acknowledgment field, making a single byte containing a sequence number sufficient for the frame header. The lack of acknowledgments means that if a frame is lost the receiver will only notice it after it has received a subsequent frame. When there are no new frames to send, the sender periodically sends a *keepalive* frame indicating the highest sequence number sent, to elicit feedback from the receiver in case the most recent data frames have been lost. The keepalive frame only uses a single byte at the header, just like a negative acknowledgment.

When a negative acknowledgment is sent, a timer is set at the receiver, equal to at least the round trip time for the link. If the timer expires and the missing frame has not been received, a new negative acknowledgment is sent and the timer is restarted. If after a specified maximum number of timeouts the frame has not been received, the scheme gives up and releases all subsequent frames that have been received in sequence to higher layers, making the "hole" in the sequence space visible to them. The number of negative acknowledgments to send before giving up is a tradeoff between reliability and maximum frame delay. Besides avoiding higher layer timeouts, by placing a limit to frame delay we can ensure that persistent losses will not delay delivery of all subsequently received frames indefinitely. This means that by selecting an appropriate retransmission limit it is possible to support higher layer protocols and applications that are loss tolerant but delay intolerant, while enhancing the reliability of the link to a certain extent. We come back to this point in Chapter 5 when we consider schemes for UDP applications.

The RLP scheme eliminates retransmission timers at the sender side, so it never retransmits a frame that was not explicitly negatively acknowledged, thus avoiding duplicates. Since it only uses negative acknowledgments and only has one byte of extra header overhead, it is usually more economical than SR variants, although it generates periodic keepalive messages. One drawback of RLP compared to SR schemes is its sensitivity to parameter settings. If SR

retransmission timeouts are too long or the window is small, its buffer array may fill up leading to frame drops, while if the timeouts are too short wasted retransmissions will occur, but in any case it will not produce wrong results. With RLP however, since the sender has no notion of a window, we must guarantee that the buffer size, the number of retransmissions and the retransmission timer length will *never* allow the buffer array to overflow. This means that the maximum frame delay, i.e. the maximum time before giving up on a frame, must be smaller than the time needed to transmit the full buffer array. If we fail to do so, the sender will replace frames that are still outstanding with new ones, their sequence numbers will be confused, and the sender and receiver will lose synchronization, leading to a link layer reset.

4.2.3 Other Schemes

The final scheme that we tested was the Snoop protocol [42], which was already described in Section 2.4. Snoop is a TCP aware link layer scheme, so it avoids any control overhead of its own relying instead on TCP header fields for its operation. It operates at the base station by retransmitting lost frames towards the wireless host after receiving duplicate TCP acknowledgments implying that loss has occurred. These acknowledgments are then suppressed to hide the loss from the TCP sender. In the reverse direction Snoop cannot retransmit faster than TCP since it has to wait for end-to-end duplicate acknowledgments to be returned from the far end of the path. It can only set *Explicit Loss Notification* (ELN) bits on duplicate acknowledgments, based on heuristic rules to determine whether a loss was due to a wireless error or not. This enables a suitably modified TCP sender at the wireless host to retransmit packets marked with the ELN bit without triggering congestion control. Since Snoop tracks TCP sequence and acknowledgment numbers for its operation, it needs to maintain separate state variables for each TCP connection traversing the base station.

The implementation of Snoop distributed with ns-2, written for LAN environments, was unusable even with the unmodified version of the simulator due to crashes, most likely due to recent changes in the simulator that were not reflected in the Snoop module. We isolated the problem to interactions between the LAN and Snoop modules, so we ported Snoop to the simpler point-to-point links, and it worked without any problems, performing as expected. We made only a few modifications to the Snoop module: we renamed the simulation objects to avoid naming conflicts, used a point-to-point link send/receive interface, moved protocol state to

a separate array, eliminated redundant LAN code and added statistics gathering support. These changes left the algorithms used by the original module intact, and did not modify its behavior in any way. A worthwhile modification would have been to replace the scheme used to maintain state for multiple TCP connections. Currently, Object Tcl hash tables are used, which means frequent invocations of the interpreter causing very degraded simulator performance.

We have also implemented a few *forward error correction* (FEC) schemes and a variant of RLP that releases received frames to higher layers in order of arrival rather than in sequence. We describe these schemes in Chapter 5 since they are used in UDP application tests. Preliminary tests with FEC schemes showed that they were inferior to ARQ schemes for TCP applications, being simultaneously less reliable and more wasteful. Out of sequence frame delivery was also found problematic for TCP: frames received by TCP out of sequence after a loss led to duplicate acknowledgments that arrived at the TCP sender before the link layer scheme had time to retransmit the missing frame. For these reasons we did not perform the full suite of TCP applications tests for these schemes.

A scheme that we considered implementing was a multiple acknowledgment variant of SR as used in the IS-54 RLP [16] and elsewhere [56, 57]. In these schemes the sender keeps track of the actual order of transmission for each frame, by maintaining a *frame transmission order* (FTO) variable that increases every time a frame is sent. The state kept for outstanding frames includes their FTO along with their sequence number. When frames are retransmitted, their FTO is also updated. The receiver returns a bit map indicating whether each frame in the window has been received or not, instead of acknowledging single frames. Besides using the bit map to advance the transmission window, the sender also identifies the acknowledged frame with the highest FTO. Any frames that are not acknowledged and have a lower FTO than this frame, must be lost, since a frame transmitted after them (hence the higher FTO) has been received. This scheme is more efficient than other SR variants [57] but more complex to implement. A simpler approach uses multiple acknowledgment bit maps without maintaining the FTO [58]. Since our main goal was not to find the absolutely best scheme for each link but to compare different approaches, we decided to invest our time in testing different types of schemes, such as the IS-95 RLP and Snoop, rather than further refining the full recovery schemes.

4.3 Simulation Results

In the following paragraphs we present and discuss TCP application performance results from extensive simulations using ns-2. Each result shown is averaged from 30 test repetitions with different random number generator seeds but otherwise identical parameters. For throughput we also include error bars depicting the variance among test repetitions: they show the mean plus and minus one standard deviation. For each link layer scheme we used the same parameters for both file transfer and WWW browsing. We set the values of link layer scheme timers based on link characteristics such as propagation delay and transmission speed, except for Snoop that estimates such values itself. We were generally conservative in our choices to avoid results that were very sensitive to simulation parameters, by, for example, using relaxed timeout values and large buffer arrays and trying to use the same values for all similar schemes. The only parameter that is not inherently link dependent is the maximum number of negative acknowledgments per frame (or, equivalently, retransmission attempts) in the RLP scheme. We used the suggested default for IS-54 of 3 retransmissions [5] in all cases, to see how it would perform under diverse error conditions. All link layer schemes were instrumented to maintain and report their operational statistics, for example amount of data sent and received, number of timeouts and retransmissions, dropped frames, and so on.

The ns-2 simulator supports a variety of TCP versions, such as the *Tahoe* and *Reno* variants [28], described in Section 2.2, as well as more advanced variants like *Vegas* congestion avoidance [59] and *Selective Acknowledgment* extensions [60]. Both Vegas congestion avoidance and Selective Acknowledgment extensions improve TCP performance under loss, but they remain unable to distinguish between congestion and wireless losses. Selective acknowledgement extensions have been previously tested with lossy links and found to be inferior to link layer schemes [4]. Since these extensions are neither common nor adequate to rectify TCP limitations, we decided to use Reno as it is very widespread and more efficient than Tahoe in dealing with losses. We did not use the TCP variant with ELN extensions supplied with ns-2 [4] as its performance was very unpredictable and, most of the time, inferior to Reno. The TCP Reno implementation in ns-2 does not simulate initial connection establishment and final connection release [26], meaning that actual TCP performance is slightly worse than its simulated version.

TCP timer granularity is another important issue as it influences the speed with which

TCP can react to losses. Most existing TCP implementations use timers with 500 ms granularity, but simulations have shown that reducing timer granularity to 100 ms improves performance under congestion [61]. Fine grained timers could improve TCP performance under wireless errors for short delay end-to-end paths, as they would allow timers to expire faster thus reducing TCP inactivity periods. However, the TCP round trip time estimation mechanism [27] intentionally uses coarse timer granularity in its low pass filter to avoid premature timeouts due to minor delay variations [61]. Therefore, very fine grained timers have the potential to cause instability [61]. In our tests we are not simulating congestion and router delays, which considerably inflate timeout estimates in real life. We thus decided to employ the more common 500 ms granularity timers, to avoid unrealistically low TCP timeout values.

The simulator also supports two TCP receiver variants. The first variant returns an acknowledgment for every received data segment. The second variant *delays* sending acknowledgments for a small interval. If more data segments arrive early enough, only one cumulative acknowledgment is sent covering all of them. Out of sequence segments however trigger generation of a duplicate acknowledgment immediately. We decided to use the first variant as delayed acknowledgments could reduce the effectiveness of TCP aware schemes like Snoop that rely on TCP acknowledgments for their operation, placing them in a disadvantaged position. We extended TCP receivers to maintain and report appropriate statistics.

4.3.1 File Transfer Performance

Figure 4.2 shows the average file transfer throughput obtained with various link layer schemes in the Cellular/LAN1 scenario, i.e. a path composed of a Cellular link and a wired LAN link with the data sender on the wired side, as depicted in Figure 3.2. Error bars indicate the mean plus and minus one standard deviation among all test repetitions. Cellular link parameters are shown in Table 3.1 while file transfer parameters are given in Table 4.1. Each line shows how a particular scheme performs under varying error conditions. The baseline is the *Default* scheme, which is TCP Reno without any link layer enhancements. Its performance dramatically drops with increasing error rates. Surprisingly, GBN style ARQ is even worse in two of the four cases. This is due to its redundant retransmissions that waste the very limited available bandwidth, prohibiting transmission of new data. However, at higher error rates even simple recovery is better than no recovery. The best performing scheme in this scenario for all but the highest loss

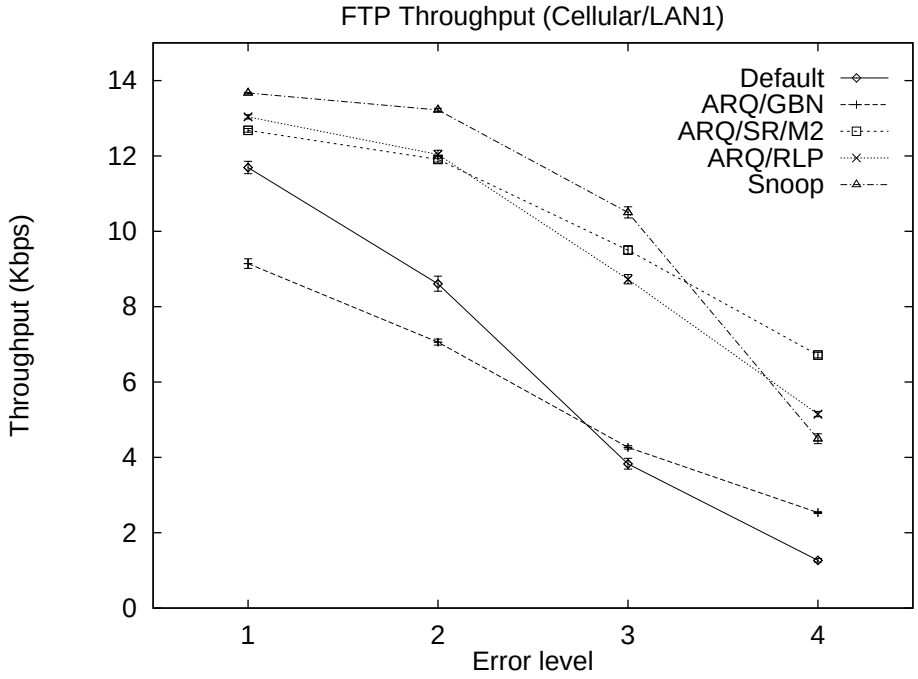

Figure 4.2: File transfer throughput over one Cellular link

rates is Snoop, with a small lead over TCP unaware schemes. Its reliance on TCP for control purposes though makes its relative performance deteriorate as the loss rate increases: as more duplicate TCP acknowledgments are lost, Snoop has to rely more on, inherently slower, timeouts for error detection and retransmissions.

SR scheme performance in all tests was ordered in terms of their complexity, with SR/M2 better than SR/M1 and SR/M1 better than plain SR. Since negative acknowledgments trigger recovery faster than timeouts, schemes that generate more accurate negative acknowledgments become increasingly preferable as the loss rate increases. For this reason, to reduce clutter in the figures we only show the performance of the most efficient SR scheme, i.e. SR/M2. The RLP scheme has a slight edge over SR/M2 for low loss rates, due to its lower per frame overhead (one byte compared to two bytes) and its very efficient negative acknowledgment only policy. As loss rates increase however, it becomes inferior to SR/M2 because its limited retransmission policy allows some losses to become visible to TCP, initiating end-to-end recovery, a situation more evident at the higher loss rate case. The overall performance improvements over the Default case are substantial: for RLP they are in the 12-305% range, depending on error rate.

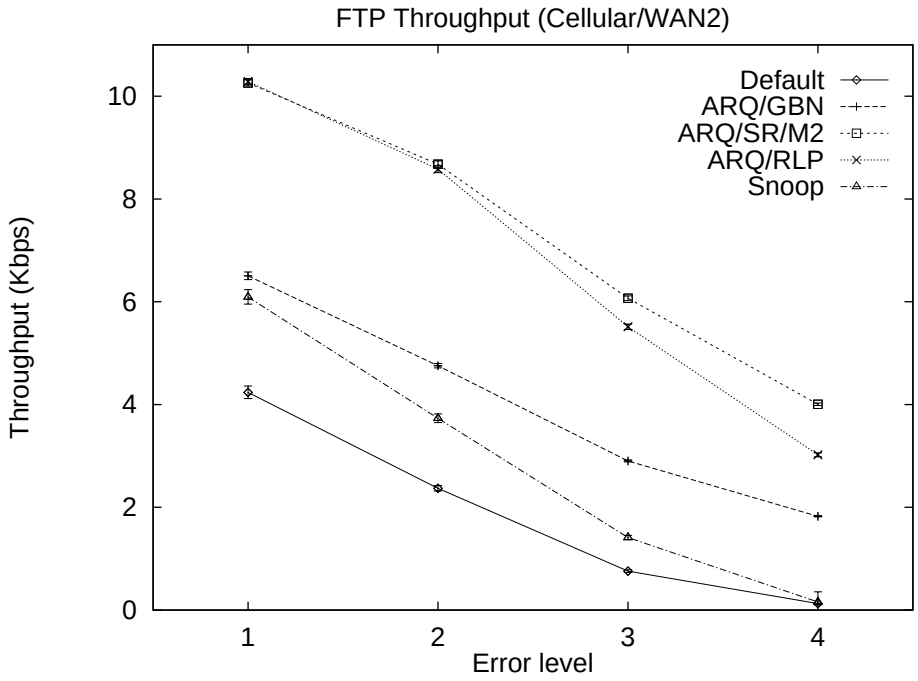

Figure 4.3: File transfer throughput over two Cellular links

In the WAN1 topology (not shown) the relative ordering of the schemes is similar, but the gains are even larger since TCP reacts even slower with longer paths.

The results from this scenario seem to verify claims that TCP awareness is required at the link layer to optimize performance [44]. For example, part of the difference between Snoop and RLP is due to the one extra byte of overhead per frame imposed by RLP, which is avoided by Snoop due to its exploitation of TCP header fields. Actually, the differences between the schemes owe more to a bias in the Cellular link error model against SR/M2 and RLP: their uniform packet loss rate means that a 2 byte negative acknowledgment (used by SR/M2 and RLP) is equally likely to be lost as a 40 byte TCP acknowledgment (used by Snoop). Under a more realistic error model or with more efficient implementations of these schemes, for example by also piggybacking *negative* acknowledgments to TCP frames, these differences could disappear. Note also that this topology is a perfect fit for Snoop, since only TCP data segments need to be retransmitted, hence base station retransmissions are adequate. Lost TCP acknowledgments are of less importance since they are cumulative, unless the loss rate is high enough to prohibit loss detection based on duplicate acknowledgments, as discussed above.

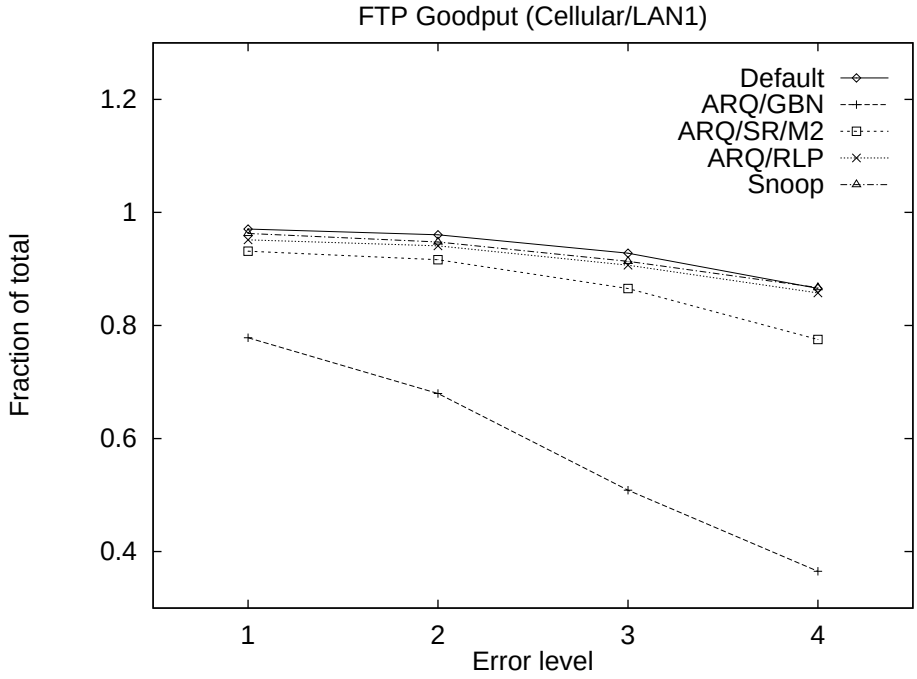

Figure 4.4: File transfer goodput over one Cellular link

The limitations of TCP awareness become apparent in the Cellular/WAN2 scenario depicted in Figure 4.3. In this case, the WAN path and the aggregate losses from two wireless links cause the Default scheme to achieve less than half its throughput in the LAN1 topology, and be surpassed by GBN. RLP and SR/M2 perform essentially in the same manner relative to each other. The worst performance among all enhancement schemes in this case however is provided by Snoop, revealing its main drawback: it only retransmits frames from the base station towards the wireless host. In two link topologies however, TCP data and TCP acknowledgments traverse base stations in both directions. Snoop can only enhance one of the two links, leaving TCP to deal with losses on the other, with disastrous results. Thus, even though Snoop does improve performance over the Default scheme, its benefits are dwarfed by SR/M2 that improves performance by 140-3000%, closely followed by RLP and its 140-2300% improvements. The results for the LAN2 topology (not shown) are very similar, indicating that the main factor determining performance is the number of wireless links in the path rather than wired path characteristics. Note also that variance is very low in all file transfer tests over Cellular links.

The most important point regarding Snoop performance in two wireless link topologies

is that its limitations are inherent to its design, because it uses TCP acknowledgments only: data segments sent from a wireless host are only acknowledged after a full end-to-end round trip time, preventing early retransmissions. Pure link layer schemes on the other hand are unaware of transfer direction and link location, performing essentially the same in every case. These results make a good case *against* TCP awareness, showing that exploiting TCP mechanisms means inheriting both its merits and its limitations. They also outline the importance of performing comprehensive tests. Previous research, by concentrating on a single topology, has only provided a limited view of these problems.

Another claimed advantage of TCP awareness is that by leveraging TCP mechanisms link layer overhead is avoided. To determine whether this overhead is significant or not, we calculated goodput, defined as total *application* data size divided by total amount of data sent by each link layer scheme [4], including both TCP and link layer overhead and retransmissions. Figure 4.4 shows goodput for each scheme in the Cellular/LAN1 scenario, which is the simplest case as only a single wireless link exists. Snoop and RLP goodput are nearly identical throughout the loss rate range, with the small difference being due to the one byte of extra header overhead for RLP. SR/M2 although relatively less efficient, is also very close, diverging only for the higher loss rates. Note however that with high loss rates SR/M2 offers *more* throughput for file transfers than either Snoop or RLP, as shown in Figure 4.2. GBN is the worst scheme in terms of goodput due to its redundant retransmissions.

PCS links suffer from a harsher, but more realistic, error model, whose parameters are given in Table 3.2. During fade periods, multiple frames may be lost in succession. It is rather unexpected then that in Figure 4.5, depicting average file transfer throughput in the PCS/LAN1 scenario, TCP does not completely collapse despite its sensitivity to bursts of losses [29]. The explanation however is simple: the error model is time based, so fade periods last for specific amounts of time rather than numbers of frames. Since TCP reacts to losses with some delay, its retransmissions have a small likelihood of encountering the same fade period. This points out an important difference when modeling bursty losses between congestion and wireless errors. Congestion appears when a router is loaded, always causing bursts of losses for sources transmitting at high rates. Wireless loss periods on the other hand do not necessarily coincide with high transmission rates, hence they may *not* cause bursts of losses. This was overlooked in previous studies that assumed that long fade periods translate to bursts of TCP losses [4].

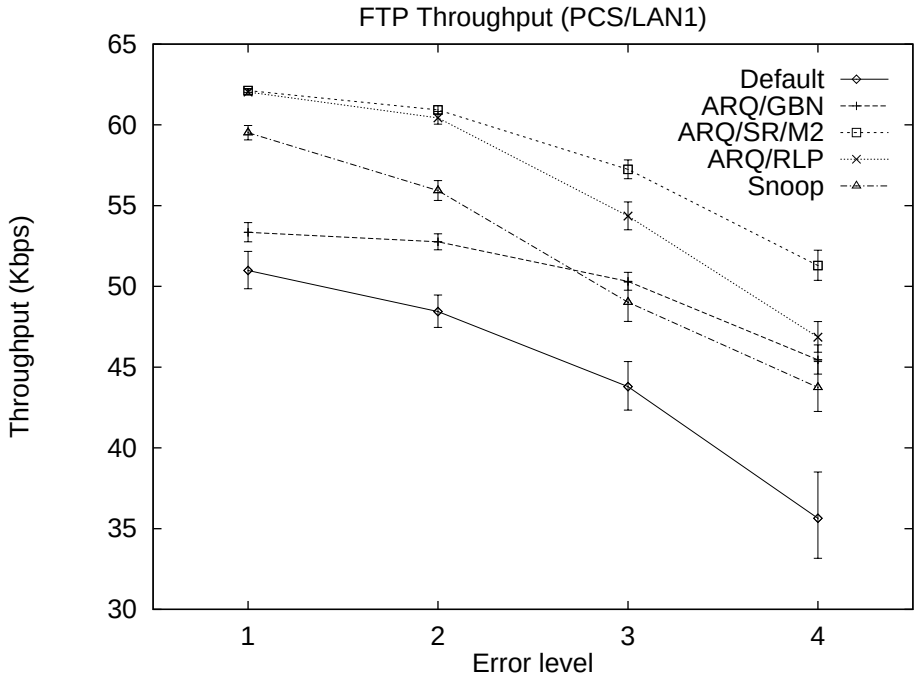

Figure 4.5: File transfer throughput over one PCS link

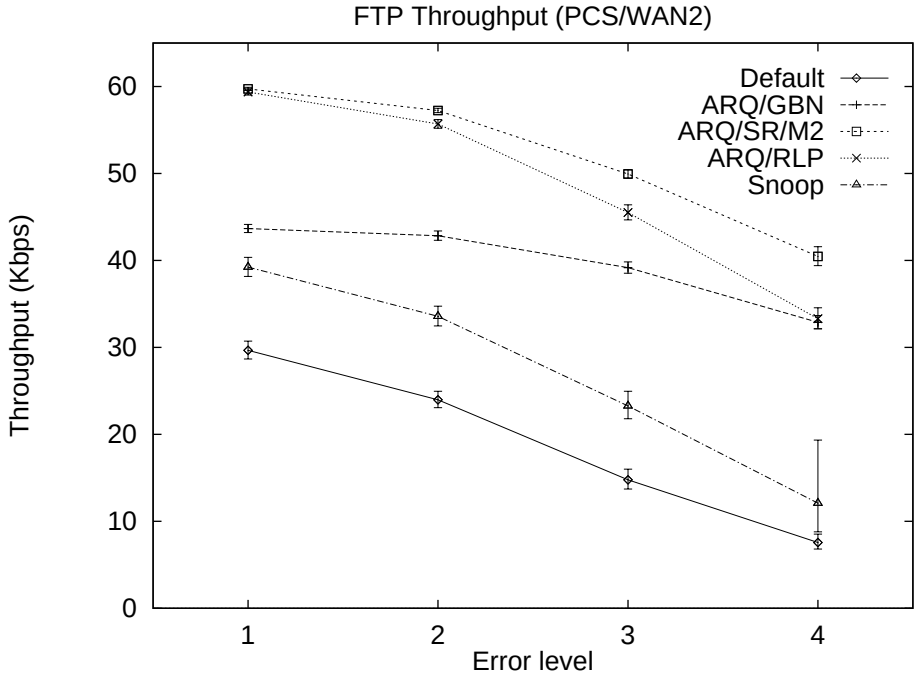

Figure 4.6: File transfer throughput over two PCS links

In this scenario, the SR/M2 scheme exhibits superior performance under all error levels, since it keeps retransmitting frames until the fade period ends. RLP on the other hand is comparable to SR/M2 with short fade periods, but becomes progressively less efficient. Longer fade periods are more likely to last long enough to corrupt either the three negative acknowledgments or the three retransmissions allowed per frame, causing more errors to become visible to TCP. Snoop performs considerably worse than both SR/M2 and RLP, the main reason being that with the time based error model, longer frames are more likely to encounter a fade period and be corrupted. Snoop relies exclusively on (40 byte) TCP acknowledgments to detect losses, while SR/M2 and RLP rely on 2 byte negative acknowledgments, hence they are notified of losses more reliably. Note that negative acknowledgments and duplicate TCP acknowledgments are generated *after* the end of a fade period, when subsequent frames arrive. This means that the limitation imposed by ns-2 to error models, namely that the two link directions are not synchronized, causes link layer error recovery performance to be underestimated, since negative acknowledgments are more likely to be lost with independent fade periods. Despite these limitations, SR/M2 improves file transfer throughput over the Default case by 22-44%, approaching the theoretical maximum very closely in all cases.

In the two link PCS/WAN2 scenario shown in Figure 4.6, the improvements are more substantial, as with two wireless links and wide area paths TCP deteriorates very rapidly, exploiting only half of the available bandwidth even under the lowest error rate. As in the Cellular case, Snoop is a relatively small improvement, due to its inability to improve TCP performance in both directions over each wireless link. The SR/M2 scheme is again the best choice for all error rates, with an advantage over RLP that increases even faster than in the one link case as fade periods become longer. The gains in this scenario are more impressive, with SR/M2 improving the throughput metric by 100-435% over the Default scheme. In addition, despite the harsh error process, its performance is again very close to the theoretical maximum for the topology. The results from the WAN1 and LAN2 topologies (not shown) are very similar. Note the increased variance compared to Cellular scenarios, which is due to the error model used for PCS links. Recovery from losses is delayed until a fade period ends, and since these periods have exponentially distributed durations, throughput performance is less predictable. Goodput for the PCS/LAN1 scenario, as depicted in Figure 4.7, varies only slightly between the schemes, mainly due to the fact that header size variations of one or two bytes are less significant for the 250 byte frames

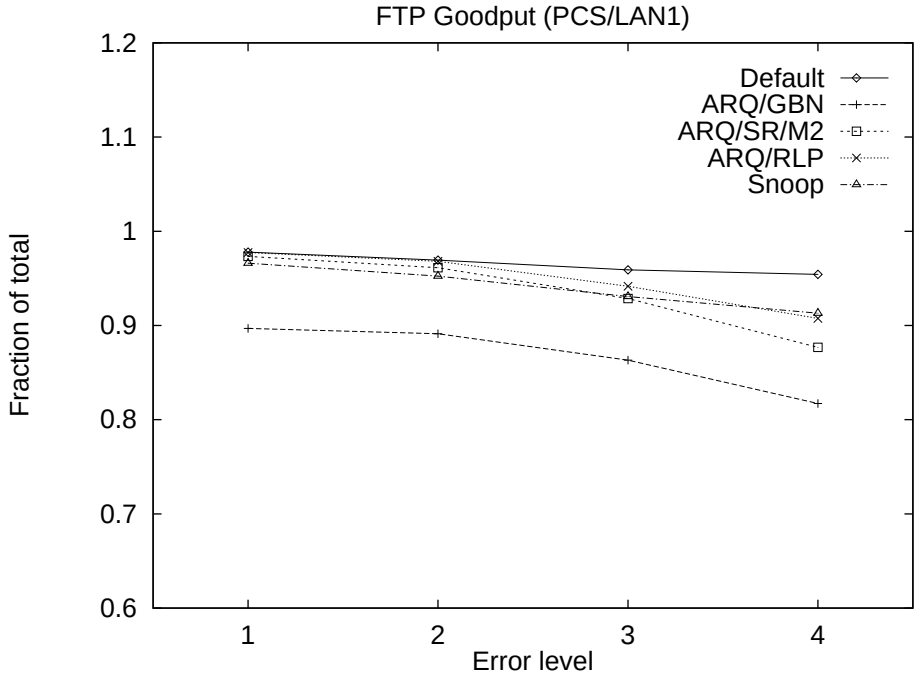

Figure 4.7: File transfer goodput over one PCS link

of PCS links. Note than in this scenario RLP exhibits *better* goodput than Snoop for some error levels, despite its additional header overhead.

The error model for WLAN links is the least harsh of all those simulated, with single bytes being corrupted in exponentially distributed intervals with quite long average duration, as shown in Table 3.3. The fact that this model advances its state only when new frames arrive, means that it is more favorable to small frames: 1000 byte TCP data segments are more likely to experience an error than 40 byte TCP acknowledgments. The average throughput results for file transfers shown in Figure 4.8 for the WLAN/LAN1 scenario are rather unexpected with respect to Default performance, which drops very sharply despite the relatively low loss rates. The explanation for this phenomenon lies partly in the coarse grained TCP timers (500 ms): losses detected by timeouts lead to underutilization of the link for all types of links, but the same period of inactivity leads to larger drops in throughput for higher speed links. As the error rate grows, timeouts are used more for error detection, hence performance drops even more.

Excluding GBN, which is as inefficient as usual, all link layer schemes lead to impressive but nearly identical improvements in throughput. As in the Cellular and PCS cases,

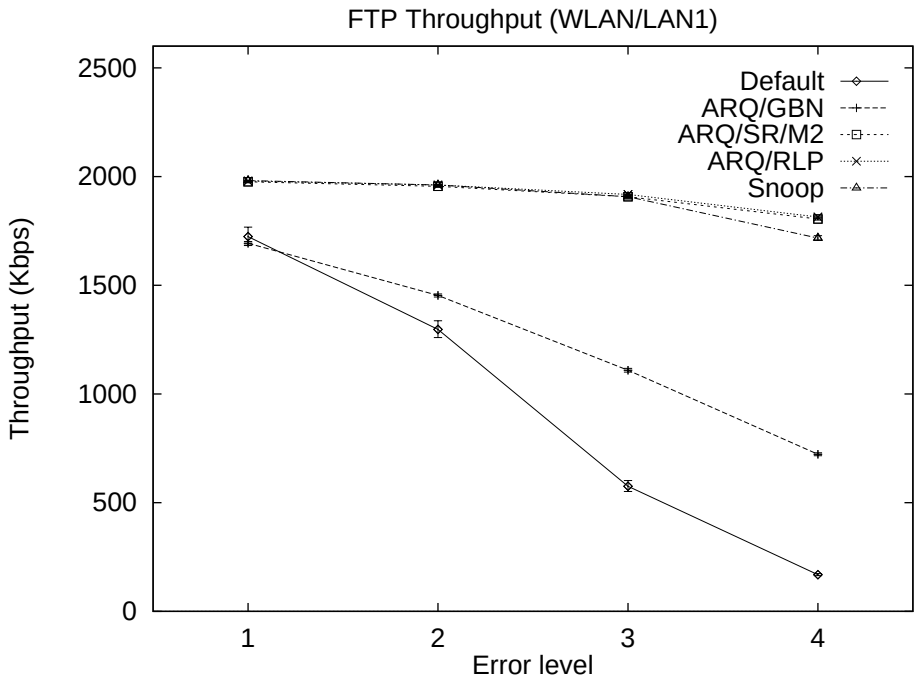

Figure 4.8: File transfer throughput over one WLAN link

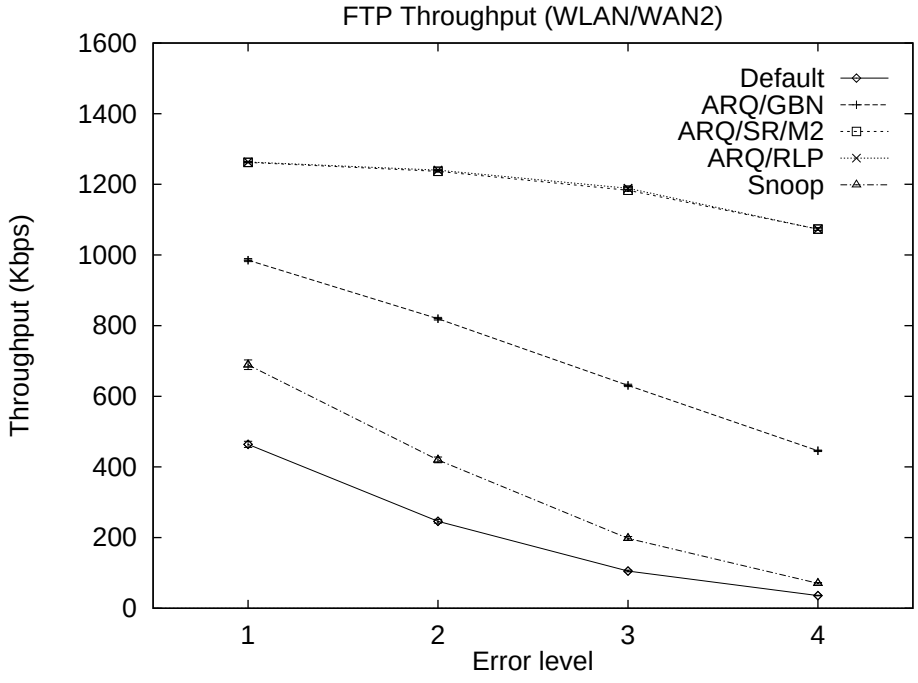

Figure 4.9: File transfer throughput over two WLAN links

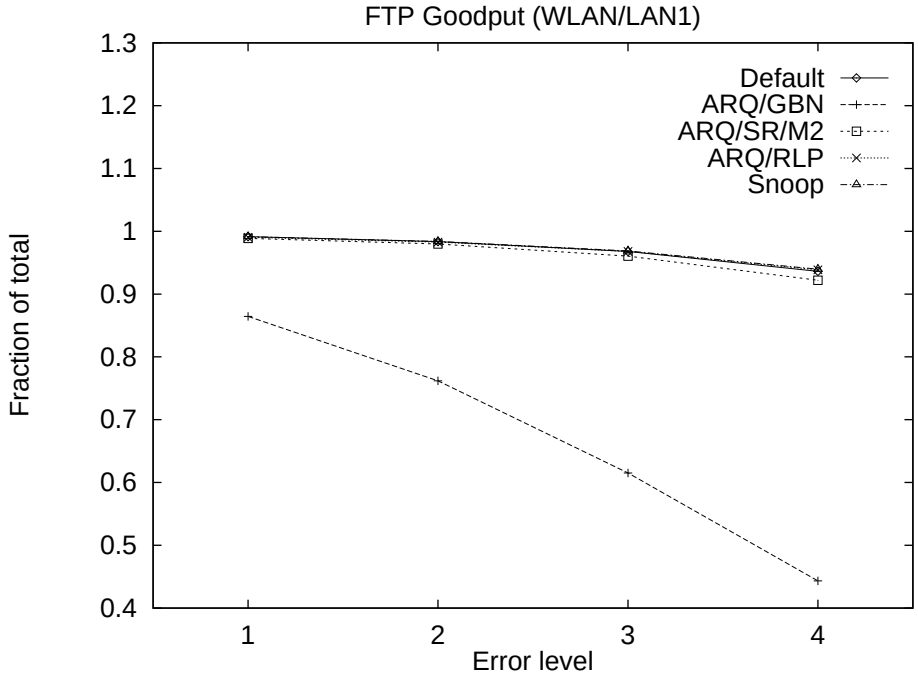

Figure 4.10: File transfer goodput over one WLAN link

SR/M2 is better than most other schemes at the highest error rates. In the WLAN case however, RLP outperforms SR/M2 by a slight margin throughout the error level range. The reason is that even the highest loss rate is low enough in this link to prevent RLP from giving up on frame retransmissions. As a result it retains the advantage it has over SR/M2 scheme at lower error rates. Both SR/M2 and RLP however offer nearly the same, and quite substantial, performance improvements over the Default scheme, falling in the 15-970% range.

In the two wireless link WLAN/WAN2 scenario, the results shown in Figure 4.9 are a mix of the one WLAN link scenario and the preceding two wireless link scenarios. The relative performance of SR/M2 and RLP is nearly the same as in WLAN/LAN1, although the throughputs achieved are lower due to the wide area path between the wireless links. Default performance is very degraded in all cases due to correspondingly higher losses and slower recovery over the WAN path. Snoop offers only small improvements over Default due to its inability to cope with two multiple wireless links on the path. Performance improvements for both SR/M2 and RLP over Default are nearly the same and more impressive than in the one wireless link case, falling in the 170-2900% range. Similar results hold for the WAN1 and LAN2 topologies

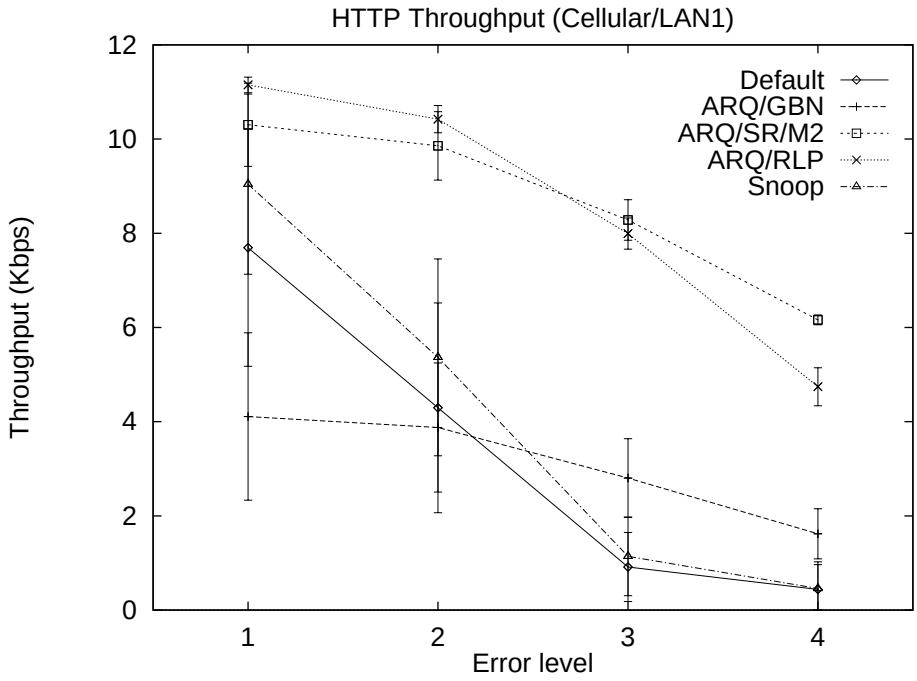

Figure 4.11: WWW browsing throughput over one Cellular link

(not shown). Due to the low error rates used, throughput variance is very limited. Goodput for the WLAN/LAN1 scenario, shown in Figure 4.10, follows the trend noted when comparing Cellular and PCS goodput metrics: the highest the frame size, the smaller the influence of the extra headers, which only represent 0.1-0.2% of total frame size for the WLAN. In this case, most schemes exhibit nearly identical goodput results, with the notable exception of GBN.

4.3.2 World Wide Web Browsing Performance

Figure 4.11 shows average WWW browsing throughput for the Cellular/LAN1 scenario. As in the file transfer case, each line shows how a particular link layer scheme performs under varying error conditions, where the Default is TCP Reno without any link layer enhancements. The error bars depict mean throughput plus and minus one standard deviation. As shown in Figure 4.1, each WWW browsing transaction consists of a sequence of requests and replies. We focus our performance analysis on reply throughput since the majority of data is transferred from the server to the client. Although each test executes for a fixed time period (2000 seconds in the Cellular and PCS tests and 500 seconds in the WLAN tests), we only consider data

transferred and time taken until the conclusion of the last complete transaction in the test. We are ignoring the transaction in progress at simulation end, as we have no way of quantifying its throughput. Note that these throughput results are *not* comparable to FTP, as in HTTP replies are separated by intervals during which requests are sent. For all WWW browsing tests we used separate TCP connections per transfer rather than one persistent connection for each transaction, reflecting the behavior of simpler HTTP versions. Our preliminary tests revealed that persistent connections moderately increase the throughput of all schemes by nearly the same amount, thus not changing their relative performance.

The results shown in Figure 4.11 for WWW browsing in the Cellular/LAN1 scenario are substantially different than the corresponding file transfer results shown in Figure 4.2. Rather than the best scheme for most loss rates, Snoop becomes the worst, offering only marginal improvements over Default. The reason is that WWW browsing is an inherently interactive application. Even though the majority of data flows from a server at a wired host to a client at a wireless host, client requests sent in the reverse direction are equally critical for good performance. If requests are not transmitted efficiently, the replies cannot be initiated, hence application performance suffers. Snoop is inherently unable to provide any improvements in the wireless host to base station direction, hence lost requests must be recovered from using end-to-end retransmissions by TCP itself. Although lost replies are efficiently retransmitted by Snoop and despite the fact that they represent much more data than requests, the net gains from unidirectional recovery are limited.

On the other hand, the performance of SR/M2 and RLP follows the exact same pattern as for file transfers. RLP starts as the best scheme at low error rates but ends up slightly worse than SR/M2 at higher error rates. The performance gains over Default for SR/M2 are 34-1300% while for RLP they are 45-980%, depending on the loss rate. These improvements are much higher than those for file transfers using the same topology and type of wireless link. The reason is that besides reduced throughput in the server to client direction, the Default scheme has to endure long idle times between replies, which are not included in the throughput metric, due to its slow recovery from lost requests. It should be clear that unidirectional file transfers, as used in most previous performance studies, are not adequate to model TCP applications in general, as interactive applications show quite different behavior due to their bidirectional nature. In addition, even when the majority of data flows towards the wireless host, for interactive ap-

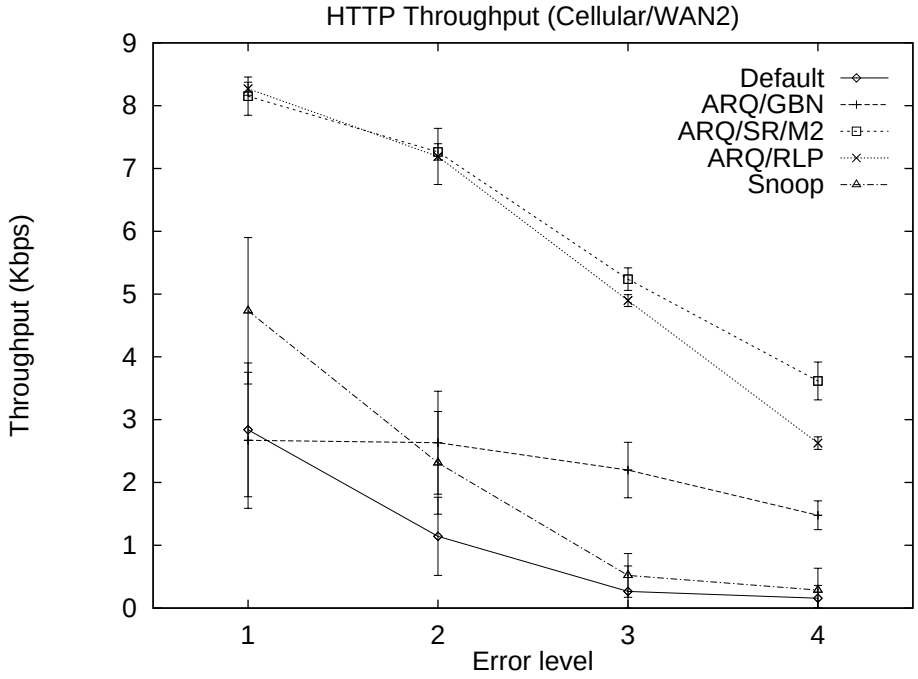

Figure 4.12: WWW browsing throughput over two Cellular links

plications both directions of the transfer are important in determining performance. Note also the increased throughput variance compared to file transfers, caused by the many short transfers used with WWW browsing: due to TCP start up dynamics, short transfers never reach the rates achievable with longer transfers. The least efficient schemes are more unpredictable since they rely more on the slower TCP timeouts.

In the two wireless link Cellular/WAN2 scenario, WWW browsing throughput, shown in Figure 4.12, exhibits a situation similar to the one wireless link case, with RLP or SR/M2 taking the lead in performance, depending on loss rate. Snoop is again insufficient to substantially improve performance over the Default case, while both Snoop and Default are further degraded by the aggregated losses on two wireless links. Snoop in particular is now unable to fully recover from losses in both the request and reply directions. The improvements over Default are even more impressive in the two wireless link case, with SR/M2 achieving enhancements of 185-2200% and RLP being in the 190-1570% range. Similarly to file transfers, the WAN1 and LAN2 topologies exhibit similar results.

With PCS links, the situation is essentially the same as with file transfers for SR/M2

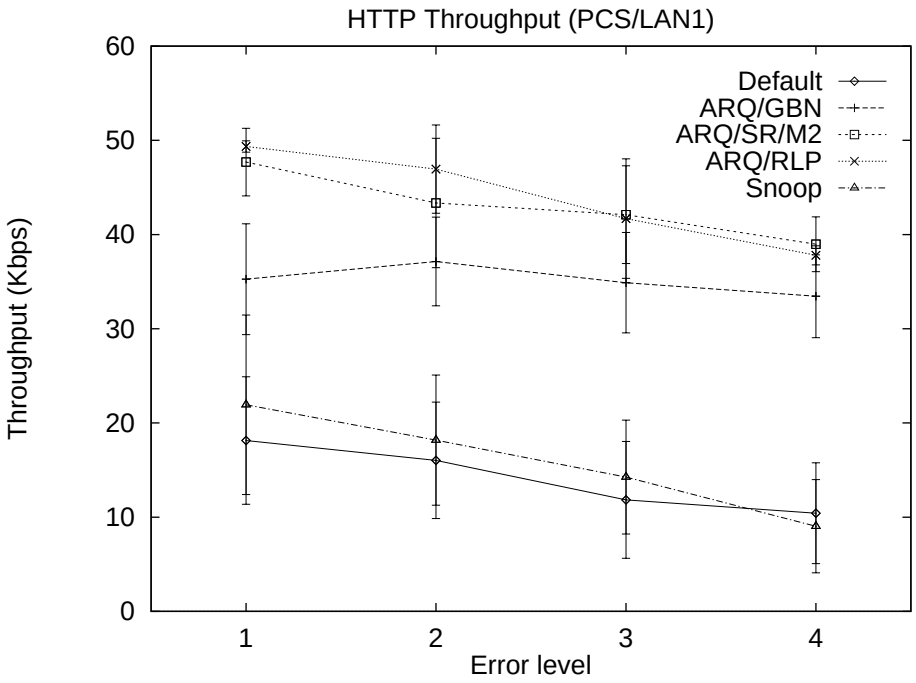

Figure 4.13: WWW browsing throughput over one PCS link

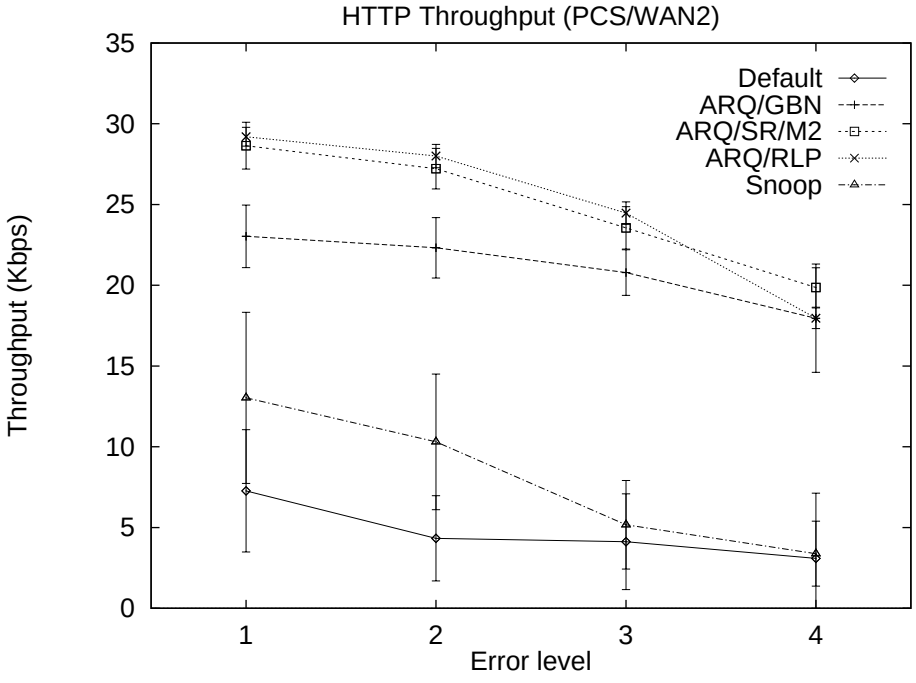

Figure 4.14: WWW browsing throughput over two PCS links

and RLP, but differs substantially for Default and Snoop. In the one wireless link PCS/LAN1 scenario, Figure 4.13 shows that the relative performance of pure link layer schemes is very similar to that of file transfers, but with smaller differences between them. Either RLP or SR/M2 are the best choices, depending on error level. The Default scheme however achieves very degraded throughput, for the same reasons explained in the Cellular WWW browsing case, i.e. increased idle intervals between replies due to inefficient recovery from lost requests. Snoop is again unable to improve the performance for requests, as they move in the wireless host to base station direction, so its performance is nearly identical to the Default scheme. The improvements in WWW browsing throughput over the Default scheme achieved by pure link layer schemes are accordingly much higher than those achieved for file transfers: for SR/M2 the improvements are 160-275% while for RLP they are 170-260%.

With two PCS links and a wide area path, as shown in Figure 4.14 for the PCS/WAN2 scenario, both Default and Snoop degrade further, for the usual reasons associated with multiple wireless link paths. The differences between the other schemes are again similar to those in file transfer tests, but slightly reduced, with RLP being slightly better than the rest for moderate loss rates. The improvements in throughput over the Default scheme are 295-540% for SR/M2 and 300-480% for RLP, again a much higher improvement than that in effect for file transfer tests. The WAN1 and LAN2 schemes further verify these observations with similar results. An interesting point is that with PCS links RLP performs slightly better on WWW browsing than on file transfer tests, relative to SR/M2. The reason is that with the small HTTP transfers often the last frame of a transfer is lost. In the absence of additional frames, SR/M2 only detects the loss after a timeout, while RLP notices it when the next keepalive frame is transmitted. As a result, RLP reacts faster and achieves better throughput results in WWW browsing tests. Throughput variance is again increased compared to file transfers, with this variability being more pronounced on the less effective schemes.

The WWW browsing throughput results for the one wireless WLAN link case of scenario WLAN/LAN1, shown in Figure 4.15, reveal nearly identical performance between the SR/M2 and RLP schemes, with RLP leading by a small margin. The difference with the file transfer case is that Snoop performs essentially at the same level as Default, which in turn performs very badly throughout the range of loss rates. Again, Snoop suffers from its inherent unidirectional limitations, failing to offer noticeable improvements over Default. As in the file

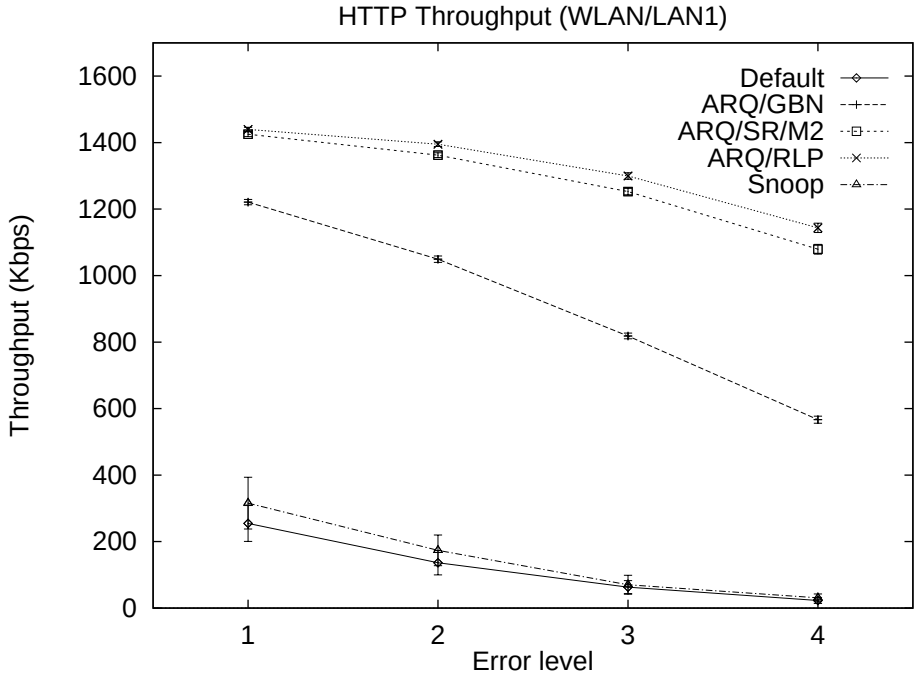

Figure 4.15: WWW browsing throughput over one WLAN link

transfer case, the Default scheme suffers more in WLAN links, as timeout initiated recovery takes roughly the same amount of time regardless of link speed but wastes more bandwidth with higher speed links. Coupled with the long delay periods between replies due to request transmissions, application throughput degrades very rapidly even under low loss rates. The best performer in this scenario is the RLP scheme, achieving improvements over the Default scheme of 465-4860%, far higher than in the file transfer tests.

Finally, with two WLAN links in the WLAN/WAN2 scenario, as Figure 4.16 shows the results are the same as with file transfers, but uniformly lower, with SR/M2 and RLP very close, and Snoop and Default lagging at a considerable distance. Compared to file transfers, the reduction in throughput in the WWW browsing tests between the LAN1 and WAN2 scenarios is much more significant. The reason is that with a WAN path between the wireless links, the time taken for each transfer is higher, for both requests and replies, since all transfers are small and cannot fill up large TCP windows, as is possible with large file transfers. This does not reflect limitations in the recovery schemes but performance limits imposed by the path and the nature of the application. Due to the uniformly lower throughput rates, in the WLAN/WAN2 case we

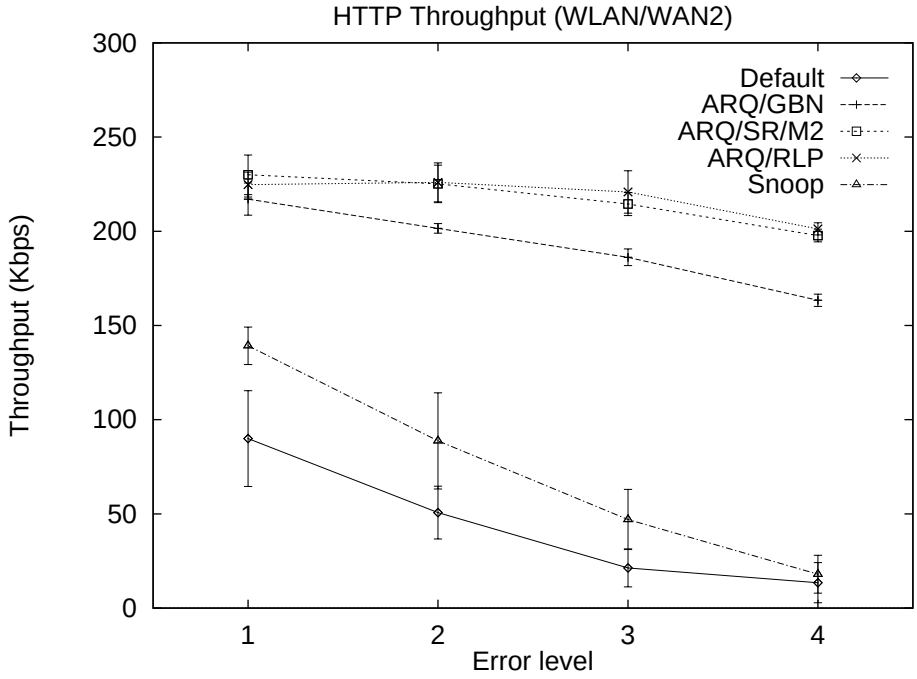

Figure 4.16: WWW browsing throughput over two WLAN links

encounter enhancements over the Default scheme that are lower than those of file transfer tests, while still being very high: SR/M2 enhances throughput by 155-1365% while RLP falls in the 150-1395% range. The WAN1 and LAN2 scenario results further verify these observations. Variance is reduced compared to Cellular and PCS tests due to the lower error rates involved.

4.4 Summary of Results

In this section we summarize our results by reviewing the performance of the various link layer schemes tested, showing how our work goes beyond previous studies and drawing overall conclusions on TCP application performance over wireless links.

4.4.1 Link Layer Scheme Performance

Regarding full recovery link layer schemes, the performance of simple variants such as GBN is clearly inadequate in most cases. The extra overhead generated by naive retransmission policies, besides limiting overall performance, wastes precious wireless link bandwidth. Among

all the SR schemes tested, the most complex one (and the only one shown), SR/M2, had a consistent edge over the rest, especially with harsh error conditions. Its additional complexity as evidenced by our implementation is low enough to make its use worthwhile in all cases. An interesting question for a future study is whether a multiple acknowledgment scheme can improve throughput enough to warrant its extra implementation and processing complexity [16, 57, 58]. Although conflicting retransmissions between the link layer and TCP did occasionally show up in our tests, revealed by the TCP receiver reporting more data received than expected, this was a problem only at the highest error rates.

The limited recovery RLP scheme avoided such conflicts by limiting its retransmissions for a specific frame. Instead of improving performance in high error situations however, it actually performed *worse* than full recovery schemes. The cause for its degraded performance was exactly that it gave up on many frames, forcing TCP to retransmit them end-to-end. Although a higher maximum retransmission limit would probably improve RLP performance, it is unlikely that the limited retransmission feature itself would make an important difference. Given the consistently superior performance of SR/M2 even under very high loss rates, in a large variety of topologies, wireless links and applications, it seems that the concerns raised on the dangers posed by conflicting retransmissions are not warranted [43]. Where RLP offers an improvement over SR/M2 is in its more economical negative acknowledgment policy that reduces both recovery time and overhead. This gives it an edge over SR/M2 under low to moderate loss rates. In addition, its keepalive feature proves very effective for small transfers as it allows many timeouts to be avoided, thus further improving WWW browsing performance.

Both full and limited recovery schemes proved very effective in improving TCP application performance. File transfer throughput increased by 10-1000% in one wireless link topologies and by 100-3000% in the harsher two wireless link topologies. WWW browsing throughput was improved even further, increasing by 35-5000% in one wireless link topologies and by 150-2200% in two wireless link topologies. Variance among test repetitions was very low for file transfers and moderate for WWW browsing which is inherently more unpredictable due to the short and variable length transfers involved. In general, the most efficient schemes also depicted the most predictable performance. Significant to impressive improvements were recorded across a spectrum of topologies, wireless links, error rates and applications, with link layer schemes of low implementation and processing complexity. Our results clearly show that

link layer recovery can rectify TCP performance problems over wireless links. More importantly, these schemes can be completely unaware of link location, TCP semantics or application behavior. The best scheme for each case depends on the error rate and error model in effect, hence no single solution is universally optimal.

The TCP aware Snoop scheme failed to improve on TCP unaware schemes, with the sole exception of file transfer over Cellular links, where its superiority was largely an artifact of an unrealistic error model that treated large and small frames the same. At best, Snoop matched the performance of other schemes (FTP over WLAN) or was slightly trailing behind them (FTP over PCS). At worst, it failed to significantly improve performance, which was the case not only with two wireless link scenarios, but even with one wireless link scenarios when WWW browsing was tested. These problems are due to its limitation of only performing retransmissions from the base station, hence an inability to recover from errors in the reverse direction. Even assuming that the majority of data flows in this direction and ignoring multiple wireless link topologies, interactive applications like WWW browsing prove that regardless of transfer sizes, *both* transfer directions matter. Furthermore, these limitations are inherent in *any* scheme that exploits TCP acknowledgments instead of generating its own. Hence, TCP awareness instead of being an advantage [44], it is a disadvantage in practice. Similarly, instead of Snoop being the best link layer solution [4], it is simply adequate for *some* file transfers and vastly inefficient for the most popular Internet application [50]. Another claimed advantage of Snoop is improved goodput due to prevention of conflicting retransmissions and reduced control traffic [4], gained at the expense of tracking the state of *all* TCP connections using the link. Our file transfer tests however reveal that the limited recovery RLP scheme provides virtually the same (with Cellular and WLAN links) or even *better* (with PCS links) goodput, hence improved goodput is *not* an inherent advantage of TCP aware schemes.

4.4.2 Limitations of Previous Studies

Previous studies have only examined TCP performance for file transfers towards a wireless host using a specific type of wireless link [4, 35, 36, 37, 39]. Their core limitation is the assumption that unidirectional file transfer is adequate to characterize TCP throughput, and therefore the performance of any application using TCP. The justification is that TCP fully controls application behavior with respect to the network and that data transfer takes place pre-

dominantly in the (wired) server to (wireless) client direction. Although the assumptions are reasonable, the conclusion is false: our results show that improving unidirectional throughput is *not* sufficient to universally enhance application performance. The more obvious problem with previous studies is that they ignored the possibility of data flowing from the wireless host towards the base station, which effectively makes their results inapplicable to paths with multiple wireless links. In these scenarios, *any* scheme limited to the base stations is inherently unable to significantly enhance performance, even for unidirectional data transfers. Modifying TCP implementations to accept ELN notifications [4], although beneficial, is no substitute for link layer recovery due to the larger time scales involved in end-to-end recovery.

This concentration on unidirectional data transfer, besides hiding the location dependencies of many proposed recovery schemes, has obscured the far more important fact that interactive applications are intrinsically more complex, as evidenced by our results. The dependence between input and output or requests and replies for these applications means that even if the amounts of data transferred in each direction are heavily asymmetric, they are both important in determining performance. This is true of both traditional interactive applications such as Telnet [52] and non interactive applications that use request/reply protocols such as POP [53]. More importantly, it is true of World Wide Web browsing, the most popular application of the Internet. Our WWW browsing tests clearly show that error recovery is required in *both* directions under *any* topological scenario in order to improve performance. For the same reasons as with multiple wireless link topologies, schemes limited to the base station are inherently unable to significantly improve WWW browsing performance.

We should also clarify an issue related to the results reported in [4]. This study claims that Snoop *considerably* outperforms TCP unaware link layer schemes exactly because of its awareness of TCP semantics. This claim is supported by testing Snoop against such a scheme, and is reproduced elsewhere by the same authors [44]. This study only measured file transfer in the wired to wireless direction. Our results with multiple wireless link topologies and WWW browsing show that TCP awareness as embodied in Snoop is actually a *limitation* in most circumstances, so this claim is not generally true. Furthermore, our tests even for similar scenarios show that TCP unaware schemes match or exceed Snoop in most cases, an apparent contradiction of earlier results that deserves an explanation. The reason for this discrepancy is the link layer scheme used for that study, which is actually Snoop without suppression of duplicate

acknowledgments [4]. Hence the scheme *is* TCP aware as it uses TCP headers and acknowledgements to detect losses. For TCP to generate duplicate acknowledgments, it must be allowed to receive data out of sequence, but since this scheme only operates at the base station this takes place automatically. To summarize, this scheme is aware of TCP semantics, releases data out of sequence, and only operates at one end of the link, i.e. a cut down version of Snoop that has all its limitations and none of its benefits. We are not aware of *any* proposed or implemented link layer scheme that fits these characteristics. Hence the claim in question is unfounded, as its definition of what a TCP unaware link layer scheme is, is self contradictory.

4.4.3 Conclusions

We decided to keep under consideration for the additional tests presented in Chapter 7 only three schemes, SR/M2, RLP and Snoop, representing the three general link layer approaches examined in this chapter. Let us now summarize our conclusions based on the results and their analysis presented in this chapter.

- Use of an efficient link layer error recovery scheme such as RLP or SR/M2 leads to considerably enhanced TCP application performance.
- Link layer schemes do not need to be aware of link location, network topology, application type or TCP semantics to be efficient.
- Reliance on TCP mechanisms and semantics for error recovery leads to degraded performance in most cases, including WWW browsing.
- Unidirectional error recovery is not sufficient to improve application performance in the vast majority of presented cases.
- File transfer is an inadequate model for predicting the performance of all interactive and many non interactive TCP applications.
- Multiple network topologies must be included in a performance study to uncover limitations such as location dependencies for each approach.
- Different error recovery schemes perform best for different links and their error models as well as for different error rates.

Chapter 5

UDP Application Performance

This chapter describes the UDP application and link layer schemes we simulated, presents detailed simulation results and analyzes their implications. Section 5.1 discusses the application chosen and its simulator implementation. Section 5.2 describes the link layer schemes we implemented and tested. Section 5.3 presents and analyzes our simulation results. Section 5.4 summarizes our conclusions.

5.1 Simulated UDP Applications

As we discussed in Section 2.2, UDP is a thin layer on top of IP, providing end-to-end best effort message delivery without any loss or sequencing guarantees. In sharp contrast with TCP that performs error, flow and congestion control, UDP does not provide any such facilities so it does not need to control the network behavior of its applications. One class of applications that use UDP are those targeted to wired LANs, where high link reliability and available bandwidth make error and congestion control unimportant. In such environments, TCP processing and state management overhead can be avoided by using the much simpler UDP. One such example is remote file access via the *Network File System* (NFS) [30], a very popular LAN oriented application. NFS is prevented from operating efficiently in a WLAN environment by the higher than expected loss rates that reveal the limitations of its slow built in recovery scheme. The requirements of applications like NFS are not substantially different than those of interactive TCP applications like WWW browsing: remote file access is based on a request and reply process, with possibly asymmetric data transfers. Indeed, NFS can be used over TCP

if needed. Since interactive TCP applications also suffer from TCP's inability to efficiently recover from wireless errors, we felt that link layer schemes appropriate for TCP would also fit LAN based UDP applications like NFS, so we did not simulate any such applications.

The other class of applications using UDP are those that have requirements not satisfied by TCP. These include applications with some real time delay requirements, for example audio and video conferencing where increasing the delivery delay beyond some point (determined by human perception factors) makes interaction impossible. Due to its mechanisms for congestion and error control TCP does not offer any delay guarantees, forcing these applications to use UDP and deal with network inadequacies themselves. IP itself does not provide delay guarantees and in practice variable delays due to congestion and network dynamics are the norm on the Internet, introducing delay variance, or *jitter*, between datagrams. If a real time application must present data to the user in an isochronous manner, it must be programmed as a *playback application* [62]: a buffer stores received data until a *playback point* and then releases them to the user. The playback point is limited by human perception requirements, so data received after the playback point cannot be used. Therefore an important metric for these applications is the fraction of the sender's data that is received before the playback point.

The other important issue for real time applications is dealing with loss. Data may be dropped due to congestion on the Internet or simply arrive after the playback point. Since end-to-end retransmissions over WAN paths would cause datagrams to miss the playback point, applications have to rely on appropriate data coding to allow the receiver to operate despite losses and late arrivals. The main issue with such coding is to include enough redundancy in the transmitted data to allow reconstruction of an adequate version at the receiver. Although the schemes used are generally designed with specific applications in mind [63], they all have to make assumptions about the loss rate in effect [64]. Settings based on expected wired Internet loss and delay become inappropriate with the additional errors introduced by wireless links. Extending end-to-end encoding to account for wireless errors is inappropriate for two reasons: first, with multiple wireless links errors accumulate increasing redundancy requirements and second, due to the dynamic nature of the Internet the number and type of wireless links traversed are not known in advance. Hence, end-to-end recovery based on coding is generally insufficient to allow real time applications to operate over wireless links, arguing for the use of link layer error recovery schemes to bring wireless link quality to an acceptable level.

It should be clear that real time playback applications have requirements that are intrinsically incompatible with TCP. Instead of complete recovery without delay constraints, they trade off complete reliability for limited delay, being both loss tolerant and delay tolerant up to a limit. Due to their inherent unsuitability for TCP, we decided to focus on these applications for UDP measurements. We describe the application we modeled in the following paragraphs.

5.1.1 Real Time Conferencing

We decided to model real time conferencing over UDP to examine what types of schemes can be used to support playback applications over different wireless links and how effective they are. A general model for these applications is multi party conferencing, with one active sender at any given time (the current speaker) and multiple receivers (the listeners). We simulated a simplified model with a single sender and receiver, i.e. one direction in a two party conversation. The general model may be viewed as the aggregation of multiple instances of our simpler model, with the constraint that only one sender can be active at any time. This simplification is *not* equivalent to using unidirectional file transfers to model WWW browsing, which was shown in Chapter 4 to be a grave mistake. In interactive TCP applications each transfer in one direction may take an arbitrary amount of time, prohibiting subsequent transfers from starting. With real time conferencing however, each transfer must complete within strict delay limits. Hence, the important question is not how much time the transfer will take but how much data will make it on time, making each transfer independent from the rest and our model sufficient to simulate the behavior of each user.

In our model, a conference participant alternates between periods of speaking and listening. We used a speech model incorporating silence suppression as the base for our implementation. In this model, shown in Figure 5.1, the sender remains in each state for an exponentially distributed period of time, with the average talk period being 1 second and the average silent period being 1.35 seconds [65]. During the talk period, the source transmits data at a *constant bit rate* (CBR) that depends on the encoding scheme used, while during the silent period no transmissions occur. This translates to transmission of fixed size packets at isochronous intervals, using the maximum packet size allowed by the wireless link tested. The average bandwidth requirements of the application are given by multiplying the peak CBR rate during talk periods by the *speech activity factor*, the ratio of talk time to total time. For the average state duration

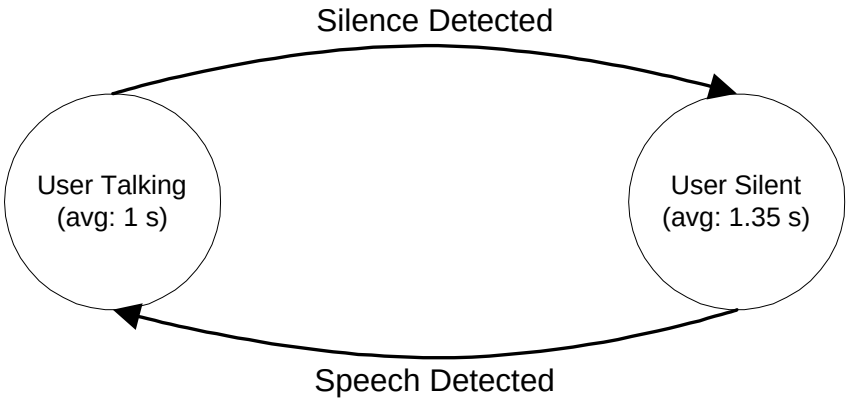

Figure 5.1: Real time conferencing model

parameters shown in Figure 5.1 this is $\frac{1}{1+1.35}$, or 42.5%, hence the average bit rate is less than half its peak value. Note that human speech patterns are *independent* of the encoding used, so this factor is valid for *all* schemes.

We chose CBR values for the talk periods based on what can be feasibly supported on each wireless link tested. Note that the peak and *not* the average rate must be less than the available link bandwidth to avoid drops, either at routers (due to congestion) or at the receiver (due to delays). For Cellular links we used 9.6 Kbps as the peak rate, which is $2/3$ of the available bandwidth, based on the very efficient cellular telephony voice encoders like that in IS-95 [5]. For PCS links we used 32 Kbps, half of the available bandwidth, which is the rate of an *adaptive pulse code modulation* (ADPCM) voice encoder as used on the DECT cordless telephony system [21] and elsewhere [65]. This data rate could also be used for low quality audio and video. For the WLAN link we also decided to use half the available bandwidth, or 1 Mbps, which can support both audio and video with adequate quality. We assumed that video is only transmitted when the user is in the talk state, to economize on bandwidth in a multi party scenario. This allows us to use the same speech activity model and parameters for all links. Table 5.1 summarizes the bit rate parameters used.

There are two metrics of interest to a real time playback application, packet loss rate and delay. Packets delayed beyond the playback point are effectively losses from the application's viewpoint, and total losses, including delayed packets, must remain below an application

Link Type	Peak Rate	Average Rate	Packet Size	Packet Interval
Cellular	9.6 Kbps	4.1 Kbps	50 bytes	41.7 ms
PCS	32 Kbps	13.6 Kbps	250 bytes	62.5 ms
WLAN	1 Mbps	425 Kbps	1000 bytes	8 ms

Table 5.1: Real time conferencing parameters

dependent threshold to allow reconstruction of the transmitted data. This threshold depends on the encoding scheme used by the application. Based on the IS-95 voice encoder, we consider losses of 1-2% to be acceptable when discussing our results, but different thresholds can be used for performance evaluation depending on application needs. Regarding delay, playback applications are interested in how many packets will arrive before the, application dependent, playback point. To estimate a playback point that would be sufficient for the majority of packets, we measured and present as a metric the sum of the average packet delay plus twice its standard deviation. We included standard deviation to account for the effects of link layer error recovery schemes on delay. For any specific application a decision can then be made on whether this playback point is within tolerance limits or not. Note that loss and delay due to wireless links should be added to loss and delay due to congestion, which is not simulated, to calculate accurate metrics for application design purposes.

5.2 Simulated Link Layer Schemes

For real time playback applications using UDP we need link layer schemes that avoid full reliability so as to limit packet delay. In addition, delivery of data in sequence is *not* required, since received data are automatically reordered in the playback buffer before use. We describe in the following paragraphs the error recovery schemes that we tested. We also discuss other schemes that we did not implement and explain our decisions. All schemes add at least 1 byte of overhead per frame, 4 bits of which store a link layer scheme identifier (see Chapter 6).

5.2.1 Forward Error Correction Schemes

The use of *forward error correction* (FEC) schemes is common in isochronous applications such as digital cellular voice telephony. The source encodes the data stream adding

adequate redundancy to allow the receiver to decode the original data despite errors, without resorting to retransmissions, thus offering a fixed limit on delivery delay. FEC schemes are divided in two broad categories: in *block coding* data are divided into independent blocks and each block is encoded separately, while in *convolutional coding* a continuous stream of data is encoded so that each block depends on a number of previous blocks [66]. Convolutional codes are often used in the physical layer, and along with the use of bit interleaving are the reason for the long frame delivery delays experienced in cellular systems as discussed in Section 2.1.

At the link layer level, most systems automatically *drop* corrupted frames, thus we have to use multiple frames in each block. This effectively rules out use of convolutional codes for our purposes since they would considerably increase the already high delays encountered in Cellular and PCS links. We decided instead to use block codes with a few frames as the block size to allow the receiver to recover from losses within reasonable delay limits. The scheme we implemented uses n frames to construct a *parity* frame that is a byte by byte *exclusive OR* (XOR) of all data frames. The original frames are sent unmodified followed by the parity frame. At the receiver, if one of the data frames is lost, it can be reconstructed by the same process, using the remaining $n-1$ data frames and the parity frame. If more frames are lost, recovery is impossible. As n grows, the level of reliability attained and the coding overhead drop, while the average frame reconstruction delay increases.

The implementation of the XOR scheme is very simple. Incoming data frames are XOR'ed one by one into a buffer before transmission. After every n frames the buffer is used as the parity frame and cleared. At the receiver, incoming frames are also XOR'ed into a buffer, including the parity frame. If only a single frame was lost in the current block, then the buffer holds a reconstructed copy of it. To avoid increasing packet delays, the receiver releases to higher layers the data frames right after using them to update the buffer. This causes reconstructed frames to be usually delivered out of sequence, unless if the last data frame in a block was lost. This allows both the sender and the receiver to use only a single buffer to hold the parity frame, regardless of n. The sender numbers all frames sequentially, allowing the receiver to detect losses and determine whether recovery is possible, adding one byte of header overhead.

Fixed block size schemes have the drawback that whenever the source becomes inactive while a block is being transmitted, recovery is delayed. For applications such as real time conferencing (described in Section 5.1) where the source alternates between silent and talking

periods, frames recovered after a silent period may be useless due to their delay. To avoid this problem, we implemented an *adaptive* variant of our XOR scheme, which we named XOR/Ad. In this scheme a block ends either when enough frames have been transmitted or when the source becomes silent. After each data frame is transmitted, a timer is set. If it expires before another data frame arrives, the block is completed prematurely by sending a parity frame. Since the XOR scheme works for any block size, the only other modification needed for adaptive operation is to enable the receiver to automatically detect block size. We used the 4 bits available in the service number byte to include a 3 bit block number and a 1 bit parity flag to each frame. Frames within a block are numbered sequentially, using one byte of overhead as in the XOR scheme. At the end of each block, the parity frame is marked with a flag, and the block number is incremented. The receiver can thus identify when a block has ended: either a parity frame arrives and recovery is attempted, or the block number changes meaning the parity frame itself has been lost. Note that the XOR/Ad scheme slightly increases the overhead and error correcting ability of a XOR scheme with the same block size, as occasionally shorter blocks are transmitted.

5.2.2 Automatic Repeat Request Schemes

Automatic repeat request (ARQ) schemes are usually considered inappropriate for real time applications due to the additional delay and jitter introduced by retransmissions. Since playback applications buffer received data until their playback point, jitter is less of an issue than the actual delays encountered. Good ARQ schemes are very economical since they only retransmit lost data, while FEC schemes rarely manage to exactly balance overhead and loss: they either add more overhead than required or use less than what is needed. We thus decided to examine whether we can use ARQ schemes for real time playback applications.

Among the mechanisms presented in Section 4.2, the Snoop scheme cannot be used with UDP applications as it is intricately tied to TCP sequence and acknowledgment numbers. The full recovery GBN and SR schemes do not provide any delay limits since they keep retransmitting lost frames indefinitely. Persistent losses may also lead to buffer overflows with protocols and applications that do not offer congestion control. When a frame is lost repeatedly, the window cannot advance beyond that sequence number, hence new packet arrivals eventually exhaust the sequence number space. When the window is full, subsequent packets are dropped, leading to losses even with a full recovery scheme. This is illustrated in Figure 5.2 which shows the

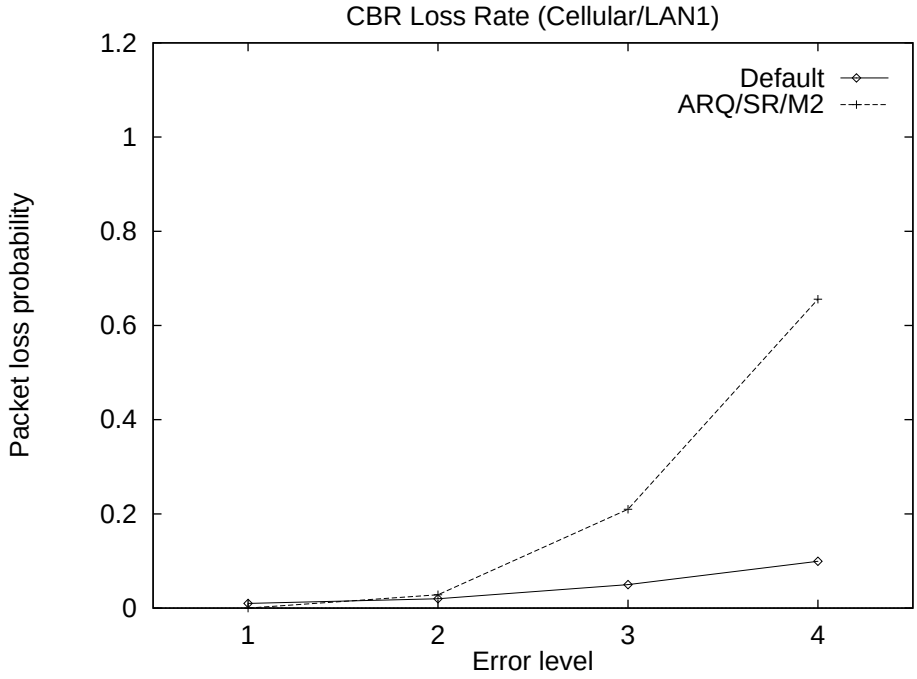

Figure 5.2: Real time conferencing loss with full recovery ARQ

loss rate of the real time conferencing application presented in Section 5.1, using either the Default scheme (no recovery) or the full recovery SR/M2 scheme, in the Cellular/LAN1 scenario. We used 127 retransmission buffers for SR/M2, the maximum allowed by our implementation. While the Default scheme suffers from the native loss rate of the link, SR/M2 is worse for all but the lowest error rates, with overflows leading to large numbers of packet drops. TCP avoids this problem since during loss periods the growth in delay causes the transmission rate to be reduced. Real time conferencing sources however transmit at a constant bit rate when active.

Persistent losses also cause delay to grow dramatically, not only for retransmitted frames but for *all* subsequent frames, which cannot be released to higher layers while an earlier frame is being recovered. This is clear in Figure 5.3 which shows average delay plus twice its standard deviation for the same scenario. Delay with SR/M2 grows to tens or hundreds of seconds even at moderate loss rates, since even one loss per window can cause many successfully transmitted packets to be delayed. These results clearly show that the most efficient full recovery scheme for TCP is far worse than no recovery for real time applications, in terms of both loss and delay. Similar results hold for nearly all other link types and topologies, hence we

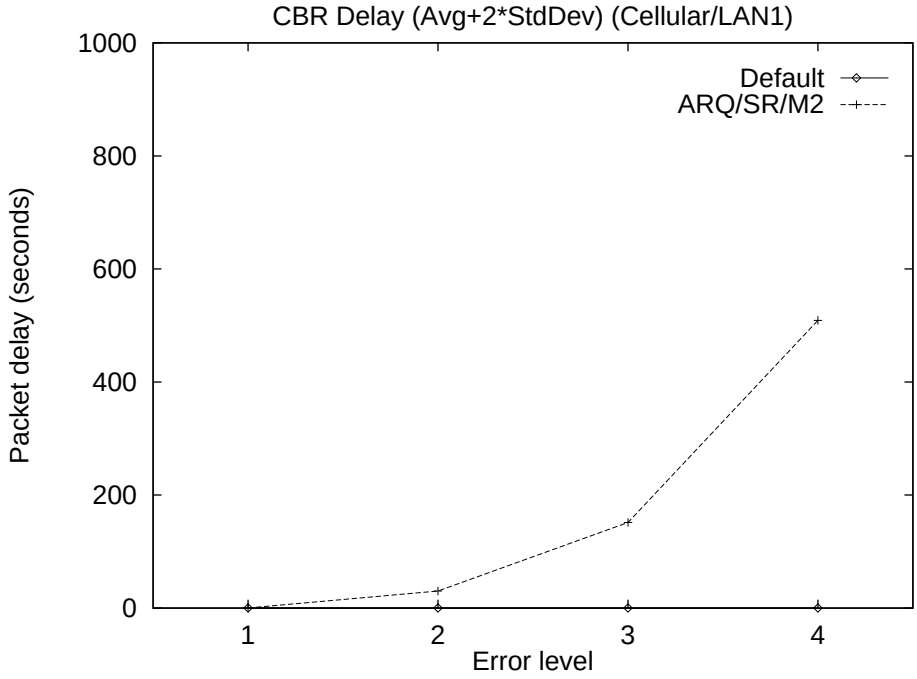

Figure 5.3: Real time conferencing delay with full recovery ARQ

decided to omit SR/M2 results for the remainder of this chapter. We instead concentrate on the limited recovery RLP scheme, which avoids the limitations of SR/M2 by dropping frames that are persistently lost so as to reduce delays and avoid buffer overflows.

As a preliminary comparison between FEC and ARQ schemes, consider our XOR scheme with $n = 9$, which means that 10% of each block is overhead. When a frame is lost, it can only be reconstructed after the block has been fully received, including the parity frame. On the average this means waiting for 4.5 additional frame transmissions to complete, so by using the frame size and speed for each link we can calculate the average recovery delay, shown in the first column of Table 5.2. A lost frame in the RLP scheme on the other hand requires one more frame to be received to notice the loss, a negative acknowledgment to be sent, and the lost frame to be retransmitted. Here however we have to add twice the propagation delay to arrive at the minimum recovery delay results shown in the second column of Table 5.2. In both cases, these results reflect minimum values since they assume no contention on the link from other traffic. As propagation delay drops and bandwidth increases, ARQ becomes more attractive: although FEC is twice as fast for Cellular, it is only slightly better than ARQ for PCS and worse than ARQ for

Link Type	FEC/XOR Delay	ARQ/RLP Delay
Cellular	130 ms	259 ms
PCS	142 ms	163 ms
WLAN	18 ms	14 ms

Table 5.2: Minimum recovery delay for FEC/XOR and ARQ/RLP

WLAN links. Hence, we decided that it was worthwhile to examine RLP with a limit of a single retransmission per frame.

Since playback applications are not particularly interested in data delivery sequence, we decided to modify the RLP scheme to release received data as they arrive, a scheme we refer to as RLP/OOS (for "out of sequence"). In this scheme the receiver does not need a buffer array since received frames are released immediately. Only a state array showing which frames have been received and which have been negatively acknowledged is used. Apart from that change, the RLP/OOS scheme is exactly the same as plain RLP, and we tested it with the same single retransmission limit as RLP in our simulations. Similar to XOR, in RLP/OOS received frames do not have to wait for subsequent lost frames to be recovered, a procedure that may not be successful anyway. Of course, lost frames *are* delayed, but this delay variance is reflected in our metric which incorporates both average delay and its standard deviation.

It may be argued that out of sequence delivery is pointless as follows: if retransmission delay is too high, then it is useless since the lost frame will miss the playback point; if it is not too high, there is no harm in delaying subsequent frames for the same interval. This reasoning is flawed however, since it only considers a single wireless link in isolation. When frames have to be transmitted over additional wired or wireless links, in sequence delivery introduces a *serialization* effect: if n frames are released together after a missing frame is recovered, then the last frame has to wait for $n-1$ frame transmission times on the next link, further increasing its delay. Even worse, with more wireless links on the path another one of these n frames may be lost, causing all subsequent frames to wait again until it is recovered. With out of sequence delivery in contrast, each individual frame is only delayed when it has to be recovered. Due to these serialization effects, we believe that out of sequence delivery is very useful for real time playback applications in general. As far as we know, we are the first to propose, and evaluate, a limited recovery ARQ scheme with out of sequence delivery.

5.2.3 Other Schemes

Instead of using FEC or ARQ in isolation, another option is to combine the two in *hybrid* schemes [67]. One possibility is to start with an ARQ scheme and add FEC overhead to each individual frame. Frames that cannot be recovered using FEC, are retransmitted. This is called a *Type-I hybrid ARQ* scheme. An alternative is to initially transmit the frames without FEC, but rather than retransmitting them unmodified after a loss, sending instead a frame of FEC overhead that when combined with the original (corrupted) frame may be able to recover the original data. This is called a *Type-II hybrid ARQ* scheme. To increase the possibility of successful decoding in a Type-II scheme, ideally each retransmission should provide additional redundancy. This is possible by using *rate compatible punctured convolutional* (RCPC) codes [68] which enable the sender to create n encoded variants of a single frame with an interesting property: the first k of these frames, for any $k \leq n$, may be used to recover the original frame, with the ability to correct more errors as k increases, so that previous (re)transmissions are not wasted. Unfortunately, hybrid schemes require corrupted frames to be available at the receiver, which is not the case in the systems we are examining.

A more interesting extension to the simple FEC schemes we implemented is to allow multiple frames to be recovered in each block. One option is to use *Reed-Solomon Erasure* (RSE) codes where for every n data frames h parity frames are generated, allowing up to h lost, or *erased*, frames to be recovered [69]. These codes are suitable for the links we examined since they assume that corrupted frames are lost. They are well suited to wireless links where losses are bursty, such as the PCS links in our study. A further extension to these schemes is to use an *adaptive* RSE code, that, like RCPC codes, allows a variable amount of overhead to be transmitted, stopping when the desired recovery level is reached [70]. These schemes considerably increase FEC processing requirements and are much more complex to implement. Error recovery delay and overhead are determined by block size, as in our XOR scheme. For RSE schemes however to be effective against burst errors, the number of parity frames per block must cover the expected duration of the bad period. Hence large blocks are needed to reduce overhead, leading to very high reconstruction delays. Thus, while RSE schemes are beneficial for end-to-end recovery in fast wired networks [69], they would introduce too much delay over each one of our wireless links to be useful to real time playback applications.

5.3 Simulation Results

In the following paragraphs we present and discuss detailed results from extensive simulations of our real time conferencing application. We used the same topologies and link parameters as for the TCP simulations in Chapter 4. Each result presented reflects the average value from 30 test repetitions with different random number generator seeds. Error bars depict average metric values plus and minus one standard deviation. Note that we first calculated the delay metric (mean packet delay plus twice its standard deviation) for *each* test, and then calculated its mean and standard deviation among *all* tests. For both ARQ schemes tested, RLP and RLP/OOS, we used the same values for the retransmission timers, based on the properties of underlying wireless links. We also limited the maximum number of retransmissions to 1 for both schemes. For both FEC schemes (XOR and XOR/Ad) we chose n, the number of data frames used to construct each parity frame, to be 8 for Cellular and PCS links and 12 for WLAN links, leading to fixed overhead factors of 11.1% and 9.1%, respectively. We set the timeout value for premature block termination in the XOR/Ad scheme to slightly more than twice the source packet interval (see Table 5.1), at 100 ms, 130 ms and 20 ms for Cellular, PCS and WLAN links, respectively. Link layers cannot normally set their timers so accurately since they are unaware of application timing, hence our results reflect best case performance. We instrumented all link layer schemes to maintain and report operational statistics, such as data sent and received, residual loss rate, and number of retransmission (ARQ) or parity (FEC) frames. We also extended UDP senders to include a sequence number and a timestamp in each packet. Our instrumented UDP receivers use this information to maintain and report end-to-end packet loss and delay statistics as perceived by the application.

5.3.1 Real Time Conferencing Performance

Figure 5.4 shows the average residual real time conferencing loss rate obtained with various link layer schemes in the Cellular/LAN1 scenario with the source at the wired side (see Figure 3.2). Table 5.1 shows the link specific parameters used. The baseline performance is given by the Default scheme, which does not enhance the link in any way. The two ARQ schemes, RLP and RLP/OOS, suffer from the same loss rate since they use identical retransmission policies and retransmission limits. In all cases they manage to keep the residual loss rate below 2%,

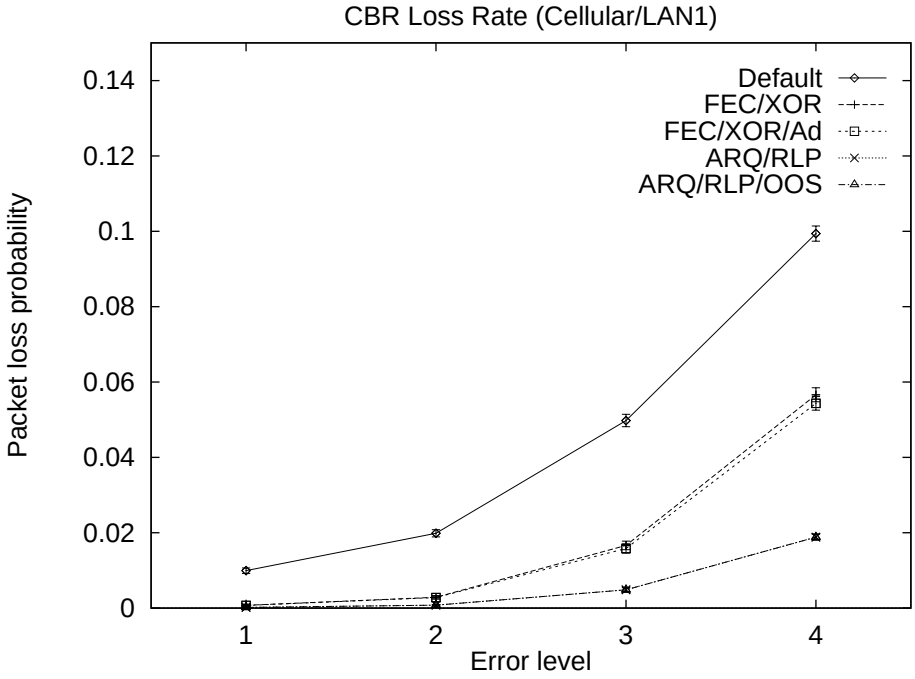

Figure 5.4: Real time conferencing loss over one Cellular link

considerably lower than both FEC schemes. Note the slight improvement with XOR/Ad over XOR due to the occasional shorter blocks. The inefficiency of the FEC schemes becomes apparent when we consider that even by introducing 11.1% of overhead, they only manage to reduce the highest native loss rate of 10% to about 6%. The problem is the fixed overhead per block which is wasted when no losses occur in a block but insufficient when more than one frame per block is lost. In contrast, ARQ schemes only retransmit lost packets, adapting their overhead to prevailing link conditions.

The corresponding delay metrics for this scenario are shown in Figure 5.5, where average delay plus twice its standard deviation are plotted for each scheme. Here, the Default scheme depicts the minimum delay possible, which, in the absence of congestion, is a fixed value without any deviation. The two ARQ schemes are here clearly differentiated. Delay for the RLP scheme dramatically increases with error rate, since after each loss many subsequent frames must be delayed during recovery, so as loss rate rises, delay follows. The RLP/OOS scheme in contrast allows out of sequence delivery, hence only lost frames suffer from increased delays. Delay rises slightly with higher loss rates since more frames need to be retransmitted,

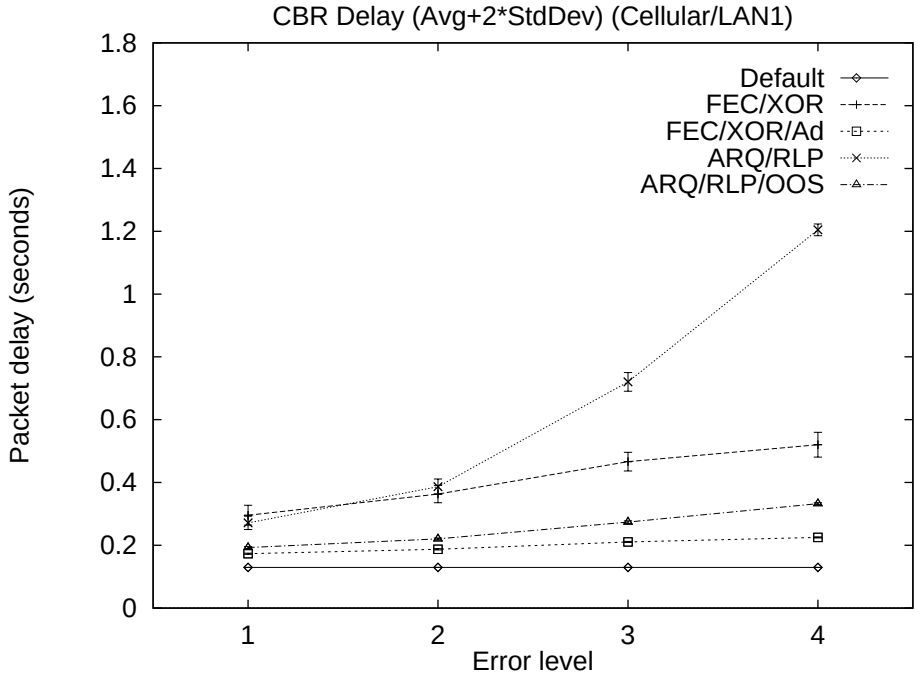

Figure 5.5: Real time conferencing delay over one Cellular link

thus increasing average frame delay. RLP/OOS thus proves to be very effective in this scenario by reducing loss rate while keeping delay low.

What is rather unexpected is that the XOR scheme suffers from higher delays than RLP/OOS, apparently contradicting our calculations about the relative delay of FEC and ARQ given in Table 5.2, which predicted XOR to be twice as fast as RLP/OOS. Examination of the detailed traces shows that while the XOR scheme depicts lower average delays than RLP/OOS, its deviation is higher, indicating that some frames are delayed too much. The reason is the on/off traffic source of the real time conferencing application: if the source enters the silent state while a block is being transmitted, the parity block will only be transmitted after the next talk period starts, thus delaying recovery. This is avoided by the XOR/Ad scheme that ends a block prematurely by transmitting a parity packet after an inactivity timeout. As a result, XOR/Ad performs slightly better than RLP/OOS, confirming the validity of our calculations.

With two Cellular links joined by a WAN path we get the Cellular/WAN2 scenario. Figure 5.6 show the loss rate in this case, with the Default scheme depicting the accumulated raw loss rate. RLP and RLP/OOS perform identically, keeping loss rates under 1% for moderate loss

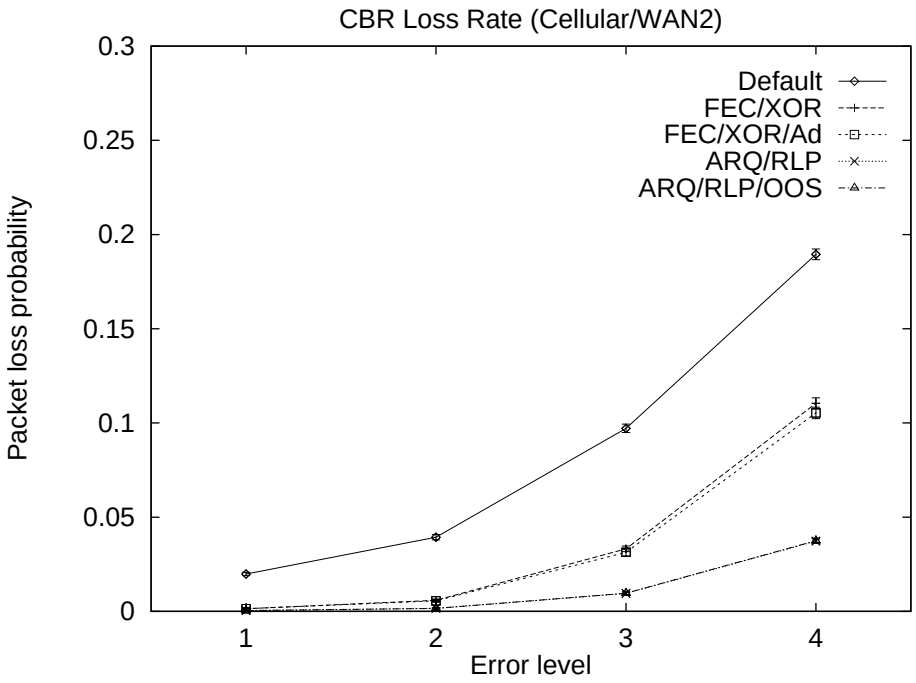

Figure 5.6: Real time conferencing loss over two Cellular links

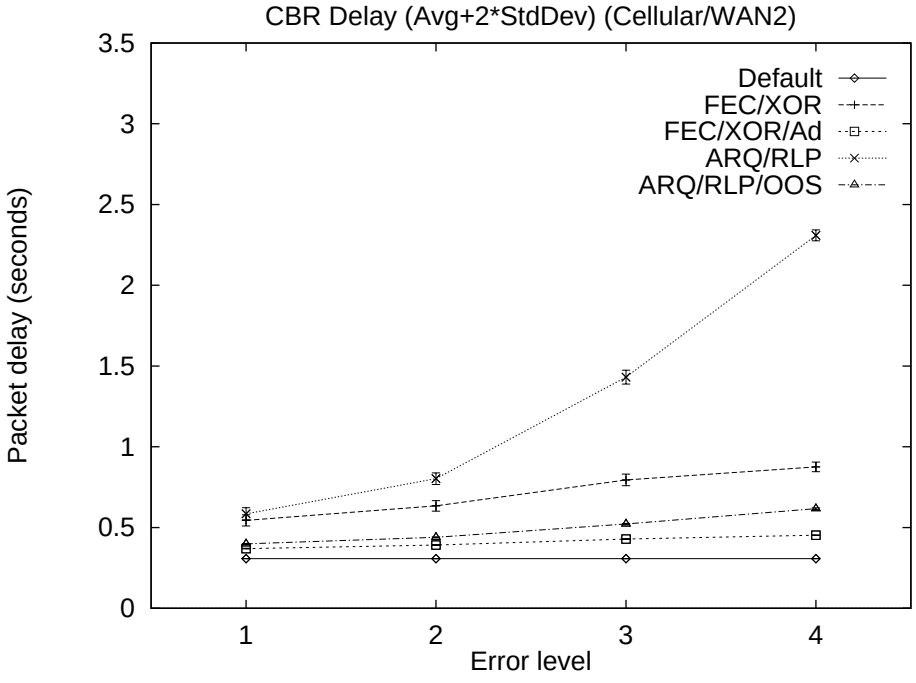

Figure 5.7: Real time conferencing delay over two Cellular links

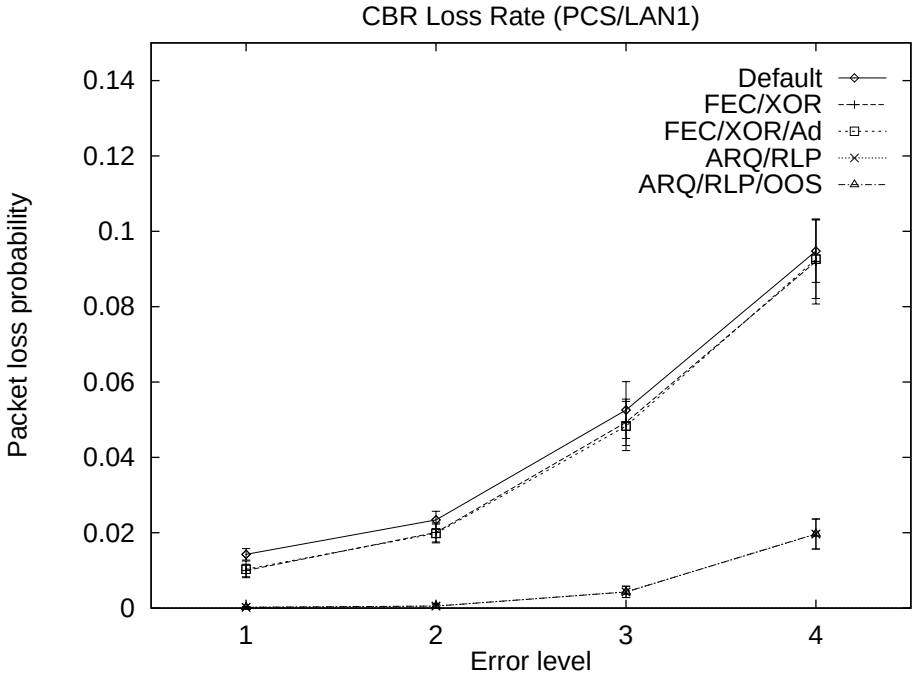

Figure 5.8: Real time conferencing loss over one PCS link

rates and only deteriorating to around 4% when the accumulated native loss rate approaches 19%. Both XOR and XOR/Ad are as inefficient as in the one link case. With respect to delay, shown in Figure 5.7, all metrics are inflated by the 50 ms delay of the WAN path. Default again depicts the lowest limit, which is twice the delay of the one link case plus the WAN delay. The extra delay introduced by the second link inflates the delay figures for all other schemes, making RLP very slow in the worst error rate. RLP/OOS lies between XOR and XOR/Ad as in the one wireless link case, due to the additional delay incurred with XOR when error recovery blocks span a silent period. Note that the gap between RLP and RLP/OOS considerably increases with two links, due to the serialization effects discussed in Section 5.2. In both topologies, loss and delay metric variance is very small. RLP/OOS again offers a very good tradeoff between reliability and delay. The WAN1 and LAN2 scenarios (not shown) further confirm these observations.

With PCS links the situation changes dramatically due to the bursty error model described in Table 3.2. Figure 5.8 shows the loss rate in the one link PCS/LAN1 scenario. Although the nominal loss rates are similar to those of Cellular links, as evidenced by the Default scheme, the two FEC schemes are basically useless, with loss rates that are nearly identical to Default.

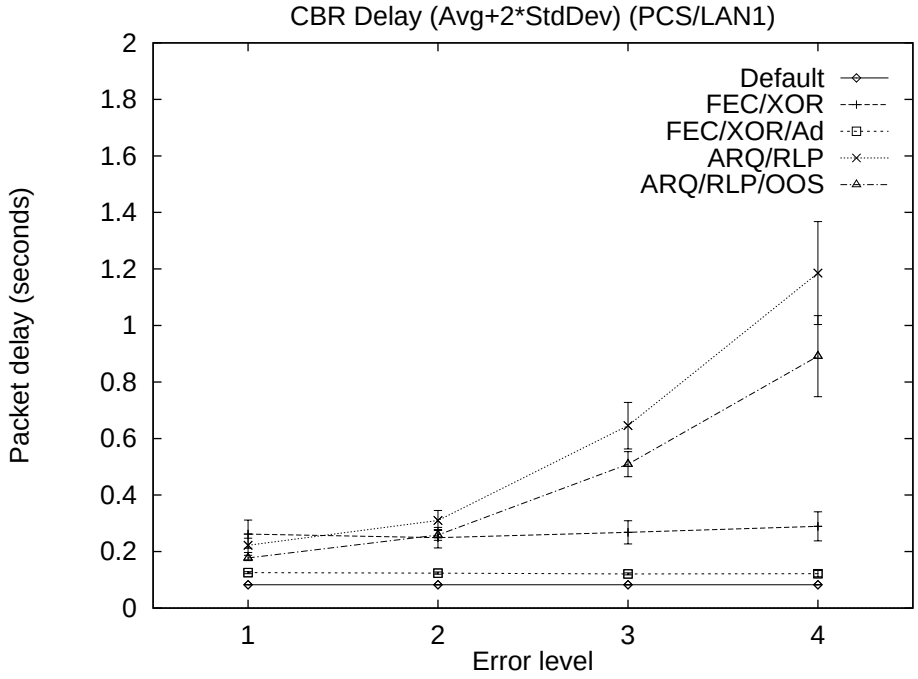

Figure 5.9: Real time conferencing delay over one PCS link

Since losses occur mostly in bursts, both XOR and XOR/Ad with *any* n are unable to recover. The sole exception is blocks that start at the end of a loss period, causing a single frame to be lost, hence the small difference with Default. An RSE based FEC scheme is better suited to this type of link, but to keep overhead at the same level the block size would need to be multiplied by h to be able to recover from loss bursts of up to h frames, increasing average recovery delay by the same factor. The ARQ schemes on the other hand achieve the same loss rate as in the Cellular case, remaining below 2% even at the highest native loss rate.

The delay results are also very different in the PCS case, as shown in Figure 5.9, but predictable given the loss results. The FEC schemes rarely manage to recover any frames, hence their delay is nearly constant across the error level range. The difference between XOR and XOR/Ad is again due to the additional delay with XOR when its error recovery blocks span a silent period. RLP is followed by RLP/OOS in exhibiting increasing delay with error rate, as more frames require retransmissions, despite the fact that RLP/OOS exploits out of sequence delivery. The reason is that during fade periods multiple frames are lost that are only detected after the fade period ends and a new frame (or a keepalive) arrives, hence increasing detection

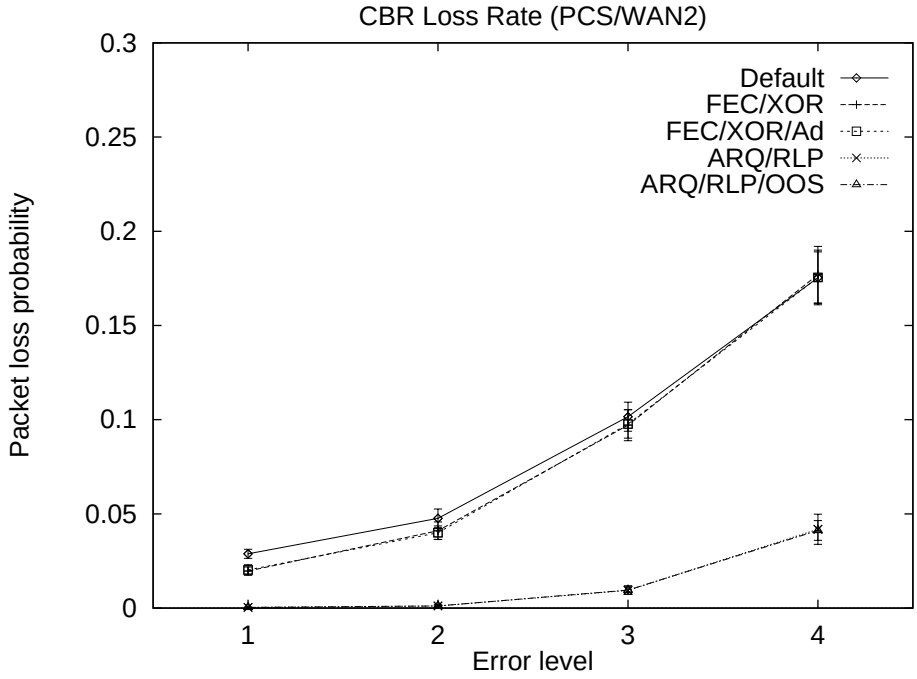

Figure 5.10: Real time conferencing loss over two PCS links

delay. In addition, when all lost frames are requested for retransmission, they must share the link with new frames, further increasing average delays. This reflects a problem inherent with this error model: frames cannot be recovered before the fade period ends, hence longer fade periods result in correspondingly higher frame delays.

With two PCS links and a wide area path, i.e. scenario PCS/WAN2, the loss results depicted in Figure 5.10 are what we would expect based on the one link results. Similar to the Cellular case, the ARQ schemes manage to keep loss around 4% at the highest loss rate and below 1% in all other cases, while both FEC schemes are nearly identical to the Default scheme. The delay metrics, shown in Figure 5.11 also closely follow the one link case: the Default and both FEC schemes exhibit essentially twice their one link delay, plus 50 ms for the WAN path. On the other hand, even though both RLP and RLP/OOS exhibit rising delays with higher error rates, the difference between them is more pronounced. This reinforces the argument in favor of out of sequence delivery made in Section 5.2, showing that serialization makes a significant difference. Delay and loss metric variance is higher than with Cellular links, but the most efficient schemes are clearly differentiated. Results from the WAN1 and LAN2

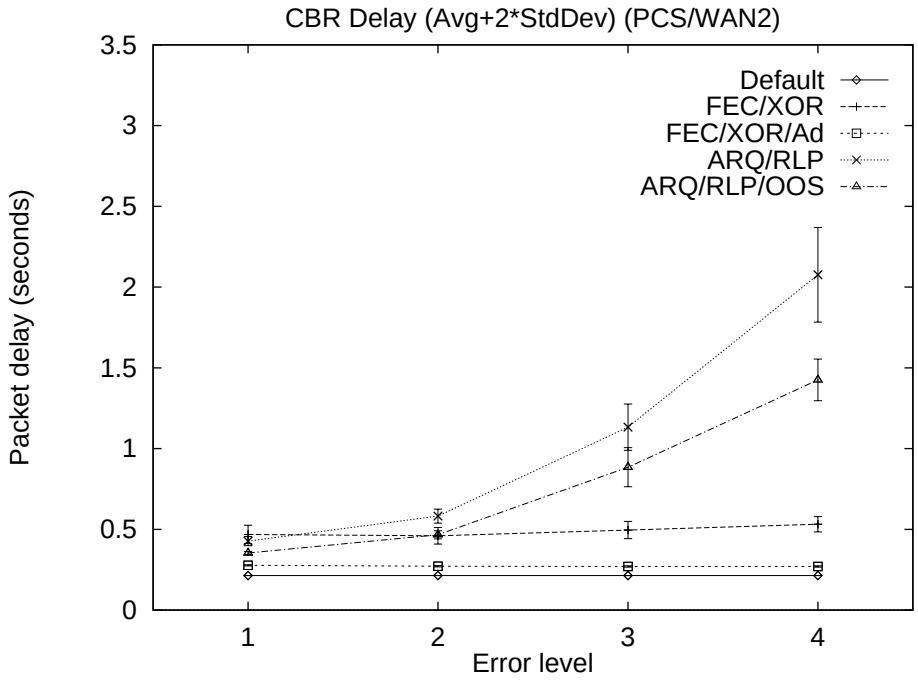

Figure 5.11: Real time conferencing delay over two PCS links

scenarios further support these observations.

Turning to WLAN links, Figure 5.12 shows the loss rate for the one link WLAN/LAN1 scenario. As noted when discussing the WLAN link parameters shown in Table 3.3, the error model is not as harsh as those of Cellular and PCS links. The Default scheme shows that the native loss rate for the 1000 byte frames used falls in the 0.8-6% range. Even though losses are rarely clustered, both the XOR and XOR/Ad schemes fail to fully exploit their 9% parity overhead, as they only reduce the highest loss rate to 3%. Both ARQ schemes on the other hand bring the residual loss rate close to zero, with nearly identical results. RLP and RLP/OOS manage to recover from nearly all loses due to the low native loss rate. Delay on the other hand, as shown in Figure 5.13, seems to verify the predictions of Table 5.2 which show ARQ to be faster than FEC for WLAN links. Due to the very low propagation delay, RLP and RLP/OOS keep delay very low. RLP/OOS is actually very close to the minimum link delay, depicted by the Default scheme. XOR/Ad is uniformly slower than RLP/OOS but better than RLP at high loss rates. In contrast, XOR suffers from much higher delays due to the long recovery interval experienced when a block is interrupted by a silent state. This problem is more evident compared

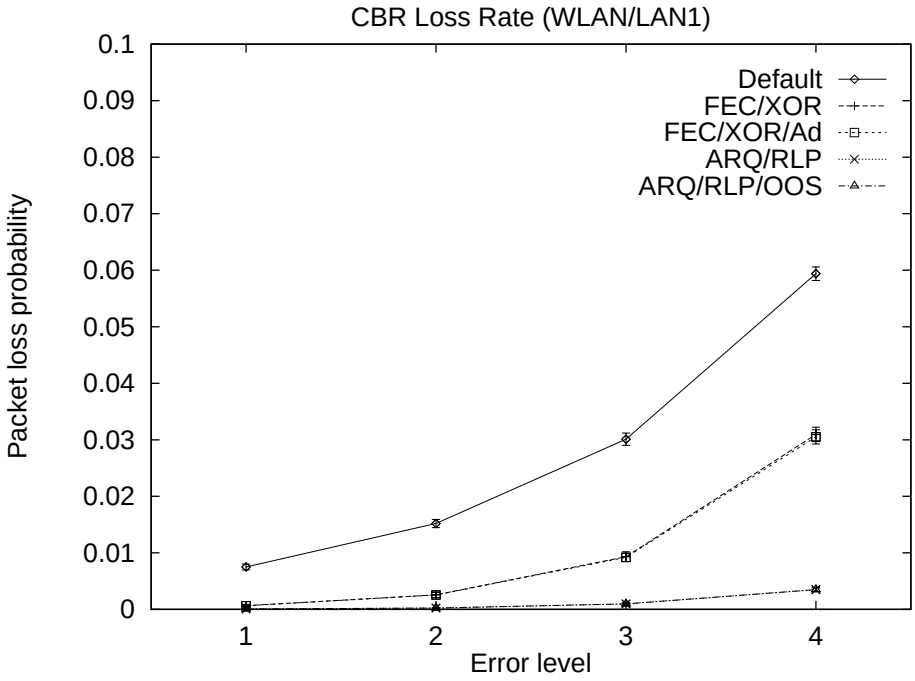

Figure 5.12: Real time conferencing loss over one WLAN link

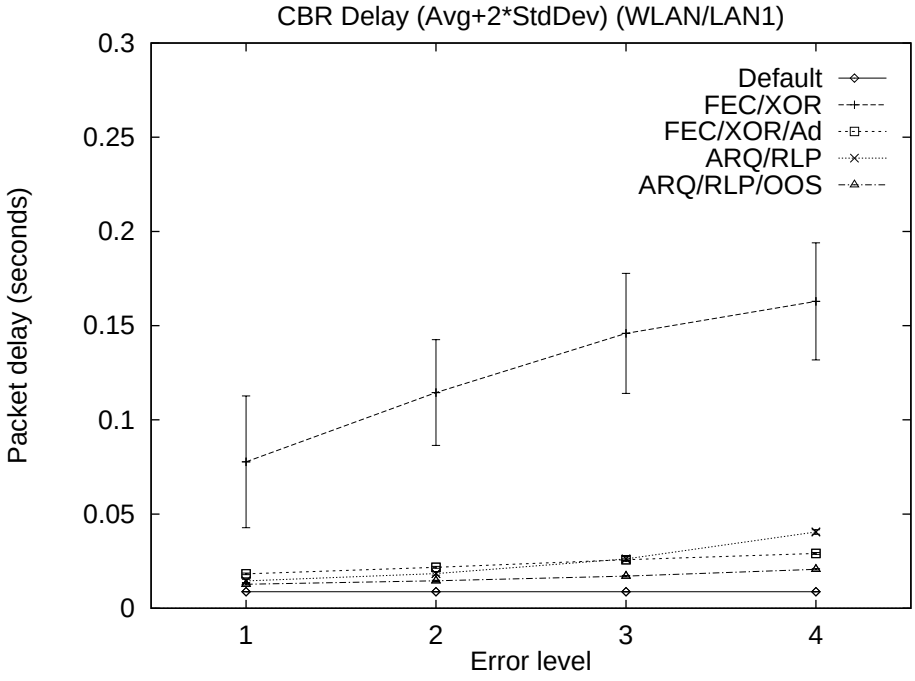

Figure 5.13: Real time conferencing delay over one WLAN link

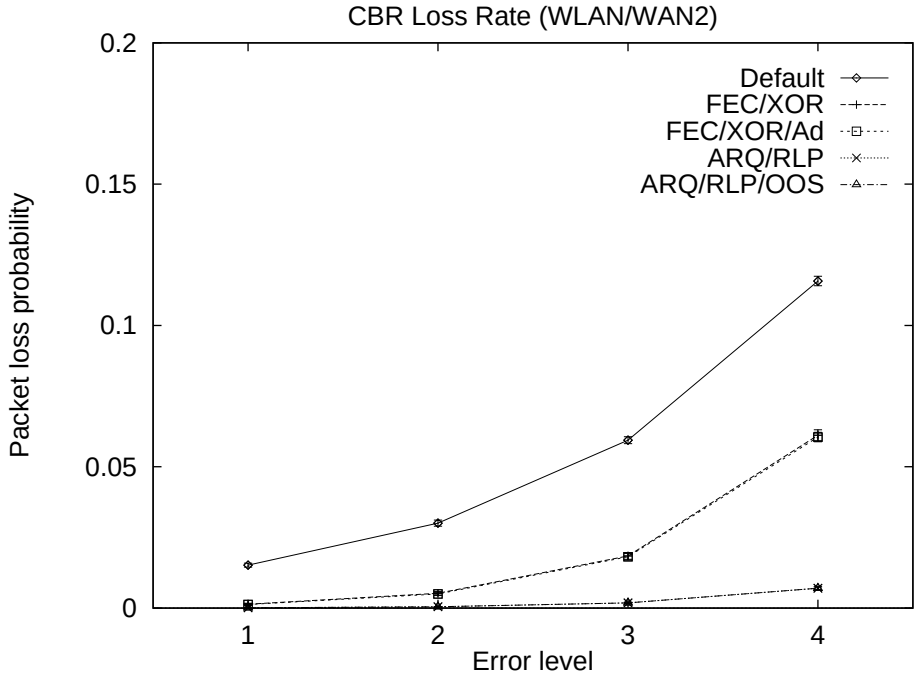

Figure 5.14: Real time conferencing loss over two WLAN links

to Cellular and PCS tests since while link speed has tremendously increased, the average silent interval, dictated by human factors, remains the same.

Finally, the two link WLAN/WAN2 scenario exhibits loss rates, shown in Figure 5.14, that are natural extensions of the one link case. Both FEC schemes remain inadequate despite their high overhead, while RLP and RLP/OOS depict residual loss rates below 0.7% throughout the error level range. The delay metrics shown in Figure 5.15 also mirror those of the one link case with RLP/OOS being very close to the Default case and clearly superior to RLP. XOR/Ad trails RLP/OOS by a small margin, while XOR is significantly slowed down by the interplay of its fixed block size and the on/off nature of the source. Similar results hold for the WAN1 and LAN2 scenarios. Loss and delay metric variance is very small in these tests with the exception of the XOR delay metric which, as explained, is adversely influenced by silent periods. Overall, in the WLAN case RLP/OOS is superior to all other schemes in both loss rate and delay terms, achieving near perfect reliability without significantly increasing Default delays. RLP is equally reliable, but lags behind RLP/OOS in terms of delay.

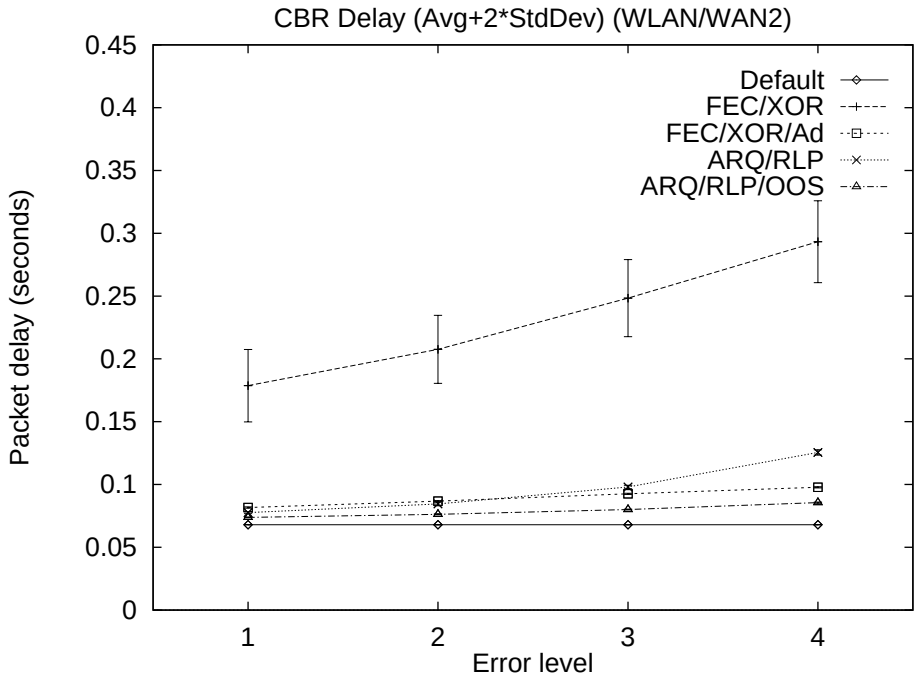

Figure 5.15: Real time conferencing delay over two WLAN links

5.4 Summary of Results

In this section we summarize our results by reviewing the performance of the various link layer schemes examined and drawing overall conclusions on the performance of real time UDP applications over wireless links.

5.4.1 Link Layer Scheme Performance

Regarding FEC schemes, the simple XOR schemes implemented revealed their limitations with all links, as the overhead they incurred did not lead to proportional improvements in residual loss rates. They were shown to be particularly ineffective for PCS links and their bursty error behavior. While FEC schemes appropriate for burst errors such as RSE coding would better fit PCS links, in order to support recovery from long bursts of errors they would have to use very large block sizes in order to amortize parity overhead. As our analysis in Section 5.2 shows, a block coding scheme that requires n frames to be received before decoding, such as RSE, suffers from average recovery delays of approximately $n/2$ frame times. As a result, large block sizes

are inappropriate for low speed links such as the Cellular and PCS links tested. On the other hand, high speed WLAN links do not exhibit the high propagation delays that motivated the use of FEC techniques with Cellular and PCS links.

Our simple XOR scheme also suffered occasionally from very high recovery delays due to its inability to adapt its block size when the source became silent. Frames recovered after such delays are nearly certain to be useless to the receiver. We thus implemented the XOR/Ad scheme that incorporated an inactivity timer to force blocks to be terminated during silent periods. Note that the reduced block size in these cases made the XOR/Ad scheme more inefficient in terms of overhead, but slightly better in its error recovery abilities. The XOR/Ad scheme was only beneficial in Cellular links with their very large propagation delays: it was ineffective for PCS and slow for WLAN links. Even for Cellular links though, the improvement in delay was relatively smaller than the degradation in residual loss compared to RLP/OOS. Note also that the XOR/Ad scheme would exhibit worse delay behavior if the inactivity timers were not tuned for the application, something not usually possible at the link layer.

Contrary to conventional wisdom but following our preliminary analysis in Section 5.2, ARQ schemes proved to be useful even for real time applications, becoming more attractive with decreasing propagation delays. ARQ schemes are very efficient, adapting automatically to prevailing loss rates since they only retransmit lost packets. They are also more effective against bursty error models like that of our PCS links. The main argument against ARQ schemes is that they are inappropriate for real time applications due to their unbounded error recovery delays. This problem is largely avoided with limited recovery schemes such as RLP by setting the retransmission limit to a low value. Our results show that even with a single retransmission limit, RLP can keep residual error rates at very low levels even at high native loss rates. To assess how effective each scheme was in terms of delay, we used a metric incorporating both average delay and its standard deviation, to estimate a playback point for real time applications that would be sufficient to receive most packets. We can thus examine whether a scheme meets the human perception requirements of an interactive application by comparing application limits with our results. This metric revealed that the RLP scheme suffered from increased delays with higher loss rates, especially in multiple wireless link topologies.

The RLP scheme supports in sequence delivery, since it was designed for TCP where out of sequence data trigger loss detection. This feature is not required for playback applications.

By delaying received frames until previous frames are retransmitted RLP causes *serialization* effects that make it very unattractive with multiple wireless links. We thus modified the scheme to release frames out of sequence and named the resulting scheme RLP/OOS. Besides reducing average delay RLP/OOS eliminates the serialization effects of RLP, delaying each frame only when it needs to be retransmitted. Our results show that this strategy is beneficial, since in two wireless link scenarios the delay gap between RLP and RLP/OOS considerably increased, making RLP/OOS very attractive for these topologies. Since RLP/OOS offers the same efficiency and reliability as RLP and considerably reduces delay even compared to the limited delay RLP scheme, it offered an excellent compromise between loss and delay in our tests. To the best of our knowledge, our study is the first to propose and evaluate limited recovery ARQ for real time applications, as well as the first one to propose and evaluate out of sequence delivery in ARQ schemes to avoid serialization delays. Note that these serialization effects become apparent only in topologies where data must be transmitted over multiple wireless links, justifying our decision to test a more extensive set of topologies than previous (TCP) studies.

5.4.2 Conclusions

Unsatisfied by the FEC schemes we tested, we decided to keep only RLP/OOS under consideration for the further tests presented in Chapter 7. We now summarize our conclusions based on the results and their analysis presented in this chapter.

- Real time UDP applications can operate over lossy links by using link layer schemes that appropriately balance reliability and delay.
- It is very hard, if not impossible, to support real time applications over slow wireless links with FEC schemes that are not very inefficient.
- Limited recovery ARQ schemes such as RLP are adequate for real time applications in wireless links with moderate propagation delays.
- In sequence data delivery introduces serialization effects that considerably increase frame delays with multiple wireless links.
- Our out of sequence version of RLP offers a very good compromise between loss and delay for real time playback applications.

Chapter 6

Multi Service Link Layer Architecture

This chapter describes a novel link layer approach that can simultaneously enhance the performance of diverse higher layer protocols and applications. Section 6.1 reviews the requirements that must be met by a universal solution to Internet performance problems over wireless links. Section 6.2 describes in detail the design of a *multi service link layer* scheme. Section 6.3 shows that the proposed architecture satisfies all of our requirements and is substantially different from other approaches.

6.1 Requirements

In Section 2.5 we formulated a set of general requirements that should be satisfied by any universal solution to Internet performance problems over wireless links. We argued that while end-to-end approaches failed to satisfy most requirements, link layer approaches looked promising. Based on our examination of TCP and UDP application performance over various link layer schemes in Chapters 4 and 5, respectively, let us review which requirements can be met by existing link layer approaches:

Location independence: Each pure link layer scheme examined offered the same performance advantages regardless of traffic direction and number of wireless links in the end-to-end path. The base station oriented and TCP aware Snoop scheme on the other hand only operated efficiently under specific topologies and traffic directions.

Adherence to layering: TCP unaware link layer schemes either matched or exceeded the per-

formance of the TCP aware Snoop scheme in the vast majority of tests. Our measurements showed that awareness of higher layer semantics is not a requirement for optimal performance and that it is actually a problem in many cases.

Easy deployment: Pure link layer schemes only need to be introduced at the two endpoints of a wireless link to enhance its performance. They can be added to the device drivers of wireless interfaces without any changes to higher layer protocols and applications at either wireless or wired hosts.

Resource efficiency: For each type of wireless link a different customized link layer scheme may be used to optimize performance, treating all higher layer data as a single flow. In contrast, TCP aware schemes like Snoop must be customized for a specific TCP variant and have to maintain per connection state.

Our measurements however also show that we cannot easily select a best overall scheme for any given type of wireless link. Some UDP applications can benefit from schemes appropriate for TCP: NFS [30] has similar requirements to other request and reply based TCP applications, thus similar link layer mechanisms can be used to enhance its performance over wireless links. Other applications however, such as the real time playback applications discussed in Section 5.1, have inherently different requirements from the network: instead of full reliability with unbounded delay, they prefer reduced reliability with limited delay. Similarly, while in sequence data delivery is essential for TCP, it is unneeded, and even detrimental due to its *serialization* effects, for UDP based playback applications. Hence, two of the requirements formulated in Section 2.5 remain unsatisfied:

Multiprotocol support: While TCP applications were best served by the SR/M2, RLP and Snoop schemes, UDP based real time conferencing worked best with the RLP/OOS scheme instead. SR/M2 and Snoop are inherently incompatible with real time UDP applications; RLP/OOS is inappropriate for TCP; RLP, although acceptable for both types of traffic, is not the best solution for either one.

Extensibility: Although the applications tested are quite representative, they certainly do not cover all existing Internet applications. Furthermore, future higher layer protocols and applications may have requirements that are as yet totally unanticipated. Since the schemes

tested are insufficient to handle our limited set of applications, they will definitely be unable to meet future needs.

We believe that the best way to also satisfy these requirements is to offer *multiple services* at the link layer: a number of link layer schemes provide their services simultaneously, with each type of traffic using the best service for its requirements [71]. Hence, a single *physical* link is transformed by the link layer into multiple *virtual* links. Each virtual link uses a link layer scheme customized to the underlying wireless link, offering a service suitable for, possibly many, higher layer protocols and applications. For example, the first virtual link could support TCP in general, the second real time UDP, with a third one offering access to the raw link. Our proposal to offer multiple services over a single physical link however raises many questions. What services will be offered? How will they be implemented? How will they share the link? How will IP datagrams be assigned to services? With these questions in mind, we can reformulate the, as yet unsatisfied, *multiprotocol support* and *extensibility* requirements for a universal solution.

Isolation: Each link layer scheme should be isolated from all other concurrently operating schemes, being ignorant of the fact that it is only seeing a fraction of higher layer traffic and sharing the physical link. This eases the programming task by allowing existing link layer implementations to be used unmodified to offer each service.

Fairness: Each type of link layer scheme may introduce an arbitrary amount of error recovery overhead which may vary considerably between schemes. While each scheme should not be restricted in how much overhead it generates, it should not be allowed to overload the physical link at the expense of other, more economical, schemes. This serves as an incentive for link layer schemes to strive for efficiency.

Extensibility: Since we cannot predict what the requirements of future higher layer protocols and applications will be, each wireless link should be able to use a variable number of arbitrary link layer schemes. New schemes should be added and obsolete schemes removed without modifications to the overall architecture or any other schemes.

All these requirements are interrelated: *isolation* eases *extensibility* by keeping each scheme independent and unaware of the rest; *fairness* eases *isolation* as it allows schemes to use any parameters they desire without having to consider their impact on other schemes; *extensibility* implies that *fairness* should be achieved for any number and type of schemes.

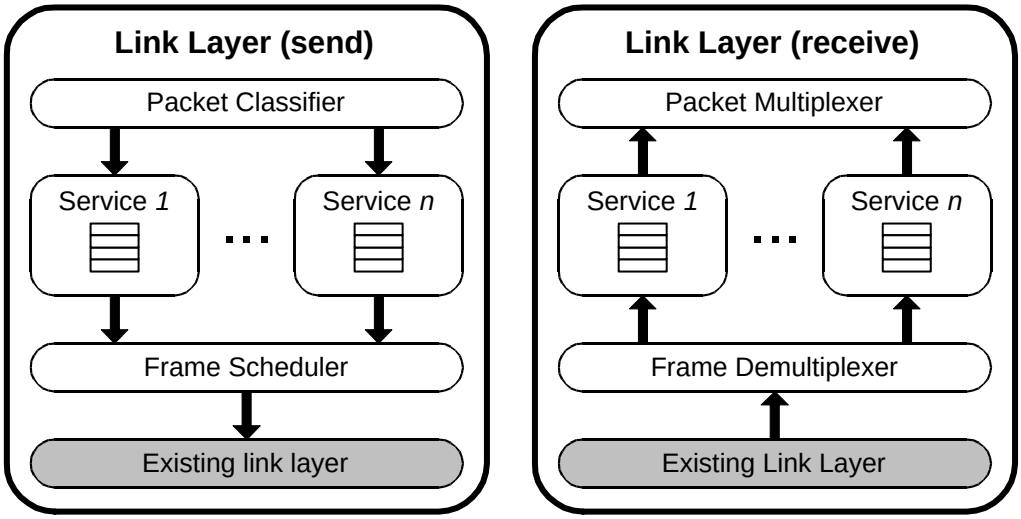

Figure 6.1: Multi service link layer design

6.2 Design

In order to satisfy all of our requirements for a universal solution to Internet performance problems over wireless links, we are proposing a model for a *multi service link layer* [72]. Figure 6.1 depicts the design of a multi service link layer, showing data flow from higher layers towards the physical layer (the *send* direction) and from the physical layer towards higher layers (the *receive* direction). In a duplex link, each link endpoint incorporates both send and receive functionality. This design provides an additional layer of services on top of existing single service link layers, as shown in Figure 6.1. To avoid modifications to adjacent network protocol layers, our scheme has a single entry and a single exit point in each direction, despite offering multiple services, with appropriate modules transparently multiplexing and demultiplexing higher layer packets and link layer frames.

As a functional overview of this design, consider how an IP datagram is sent over a wireless link using our scheme. The datagram (or packet) is passed by IP to the link layer of the wireless link for transmission, but instead of using the existing single service link layer it is intercepted by our module. A *packet classifier* examines all packets and passes them for processing to the appropriate *service*, based on packet header fields. Each service is implemented by a link layer scheme that best fits the requirements of a specific type of traffic. The service chosen

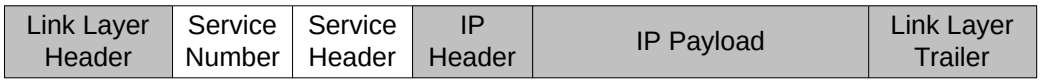

Figure 6.2: Multi service link layer headers

processes the incoming packet as if it was the only service operating on this link, performing any service dependent actions. For example, an ARQ scheme may store the packet into retransmission buffers, add sequence and acknowledgment numbers to its header and update timers, while a FEC scheme may update its coding state and transmit new parity frames. Eventually, the service outputs the packet as one or more frames which are buffered by the *frame scheduler* in per service queues. The scheduler adds a service number to the header of each frame and decides which service should transmit next every time the link becomes available, using a *fair queueing* scheme [73] and a table of service *rates* showing the link share allocated to each service. The earliest buffered frame from the chosen service is then sent to the *existing link layer*, which may execute a MAC protocol and add a common link header and trailer to each frame before passing it to the physical layer for transmission. For example, an ARQ based service that encapsulates each IP datagram in a single link layer frame will produce frames that will be transmitted in the form shown in Figure 6.2.

When a frame is received by the existing peer link layer, if it is not damaged (in which case it is dropped by most links) it is stripped of existing link headers and trailers and passed up the protocol stack. Instead of reaching the IP layer, it is intercepted by our scheme and passed to the *frame demultiplexer*. The demultiplexer looks at the service number added by the scheduler to each frame and passes the frame to the corresponding service for receiver processing. Note that service numbers do not have global significance, they simply identify one of the services available on this link. The service chosen performs any service dependent actions, again as if it was the only service operating on the link. For example, an ARQ scheme may add the frame to its receiver buffers, strip its header, update its state and generate positive or negative acknowledgments, while a FEC scheme may decode parity frames to recover missing data. Eventually, the reconstructed, and possibly reassembled, packets from each service are passed to a *packet multiplexer* that simply releases them to the IP layer as they are received.

From this description we can see that most elements of the design are independent of

the wireless link and the number and type of services offered, with the exception of the service modules themselves, which are in turn unaware that multiple services are offered. This splits the implementation of a multi service link layer in a *link dependent* and a *link independent* part: the link independent part can be reused for any type of link, while the link dependent part is customized to underlying link properties. Since each service is operating in isolation however, existing schemes can be used to provide services with minimal modifications, thus allowing a complete multi service link layer to be composed from existing code. In the following sections we describe in more detail each component of our design.

6.2.1 Classifier

The classifier routes incoming packets to link services, matching higher layer requirements with the schemes available on the link. The basic task of the classifier is to examine a number of header fields for each packet and use a *lookup table* to match their values with appropriate services. This may be implemented in a generic manner by using a *hashing function* to map these header fields to the lookup table entry pointing at the appropriate service. Packets not matched are mapped to the default service which offers direct access to the existing link layer. Thus, in the absence of any authoritative knowledge on how to treat a packet, it receives the same service offered by the underlying single service link layer. Since the scheduler ensures that services generating error recovery overhead do not take away link resources from the rest, the default service is not disadvantaged in terms of bandwidth. The key issue for the classifier is setting a hashing function and a lookup table. These should be set by an external administrative module that has an understanding of higher layer header fields (for the hashing function) and available link layer schemes (for the table). The set of services may be changed at any time by modifying some table entries. We defer a detailed discussion of how the administrative module chooses a hashing function and table entries until Chapter 8.

The classifier can optionally calculate the bandwidth shares allocated by higher layers to each service, by simply measuring the total size of packets passed to each service over an appropriate time interval. This works whether higher layers perform packet scheduling or not. If higher layers are aware of the existence of multiple services they can directly set the service rates for the scheduler, so such measurements are not required. If however they treat the link layer as a single service one, the amount of data passed for transmission for each type of traffic reflects

their decision of how the link should be shared. Therefore, the classifier can measure the desired service rates and periodically set the service rate table used by the classifier so as to reflect higher layer choices. In the following we will assume that for higher layers that do not set the service rates explicitly, these are set by the classifier so as to *exactly* match the measured bandwidth share allocated to each service. By setting the service rates in this manner the classifier remains independent of the actual services offered and their properties. In addition, services are protected from each other since regardless of the amount of overhead they generate their service rate remains the same.

A final task for the classifier is to pace incoming packets so that their *nominal* transmission time elapses between them. After receiving a packet from the IP layer, the classifier will only accept a new one after a time interval determined by dividing packet size (including any fixed link layer overhead) by link transmission speed. In current systems, the IP layer is allowed to pass one or more packets to the link layer as long as hardware buffers are available. When the buffers are full, the IP send call is blocked, until a hardware interrupt shows that buffers are again available. Since our module lies between IP and the existing link layer, these blocking calls and interrupts will be handled by the scheduler, with the classifier blocking IP send calls instead (see Figure 6.1). To improve performance, the classifier may instead only block IP send calls every few packets for the nominal time interval required to transmit all of them, starting at the time the first one was passed to it. The *actual* transmission time for a packet is generally different than this nominal time due to additional per service overhead.

IP send calls must be blocked so that data will be buffered during periods of congestion at the IP layer and not at the link layer. This is required because the actual size of IP queues is used by many schemes for congestion management and packet scheduling. Regarding congestion management, *drop tail* gateways drop packets when the queue is full, while *random early detection* gateways [74] measure the average queue size and drop packets even before congestion becomes persistent. These losses are used to notify TCP about network congestion, as discussed in Section 2.2. Packet scheduling in the *class based queueing* scheme [75, 76] also uses queue sizes to determine which packet should be sent next. Therefore, we force the IP layer to transmit only at the rate that a single service link layer would permit it to.

The classifier module is independent of both underlying link properties and available services. Its only configuration parameters are the number of services offered (used by the clas-

sification and measurement functions) and the nominal bandwidth and amount of fixed link layer overhead of the link (used by the IP blocking function). These may be set by the same external administrative module that sets the hashing function and lookup table used by the classification function.

6.2.2 Services

Each service offered by our multi service link layer is conceptually a separate entity that, to all intents and purposes, is functionally identical to a single service link layer using the same mechanism. While the service may be customized to the underlying link, it is not aware of the classifier, scheduler and the rest of the services. Each service has single entry and exit points in both the send and receive directions (see Figure 6.1), but instead of being directly connected to the IP and physical layers, it is instead connected to the classifier, scheduler, multiplexer and demultiplexer. As a result, a *virtual private* path exists between the peer services at each link endpoint. The internal operation and external semantics of each service are completely open to the designer: the service may use buffers, timers, encoders and decoders, generate arbitrary amounts of overhead in the form of parity and retransmission frames, and add any headers required for its operation to each frame.

Since services are isolated from each other by the scheduler, each one can only exploit the bandwidth allocated to it by higher layers, without however knowing how much this bandwidth is. Services must transmit all their frames, whether data or overhead, through a frame scheduler that limits their share of the link to prescribed amounts when there is contention on the link, regardless of service semantics. As a result, similar to single service link layers, inflation of the data stream with overhead leads to decreased data throughput. This is an incentive for services to optimize their mechanisms so as to achieve their goals in the most economical manner. When the load is reduced of course, services can exploit the available bandwidth to drain their internal buffers and queues.

Only a few changes must be made to an existing link layer scheme to make it suitable for multi service operation. The mechanisms used to communicate with higher and lower layers must be stripped of their hardware dependencies and modified to use a common set of send and receive calls to communicate with the adjacent modules shown in Figure 6.1. Note that these calls are service and link independent: all services use the same calls to access the link indepen-

dent modules of the multi service link layer. Conversely, a new service may be implemented for a link without having to deal with hardware and operating system specific calls to adjacent layers, using instead those common calls for input and output purposes. Services only add to each frame any private header fields that are needed for their own operation. The scheduler adds its own service identifier field to all frames to allow the frame demultiplexer to distinguish services on reception. Other headers and trailers are added to all frames before transmission and stripped on reception by the existing link layer scheme (see Figure 6.2). Thus, services are insulated from the multi service infrastructure and the underlying link. Note that, in the same manner as with single service link layers, frames passed by a service to the scheduler for transmission cannot be cancelled afterwards. The scheduler can accept multiple frames from each service, storing them in its own per service queues until they can be transmitted.

As in any pure link layer scheme, services are not aware of the type of higher layer traffic they are carrying. The classifier is instead responsible for matching higher layer requirements with services. The services are also not aware of the existence of other services since the frame scheduler ensures that they are isolated in terms of bandwidth and takes care of labeling frames with service identifiers. Multiplexing services over a single link however means increased round trip times. For example, acknowledgment frames for an ARQ service may be delayed due to contention from other services, hence the retransmission timers must not be as tight as they would be with a single service over the link. To avoid premature timeouts, services may adapt their timers dynamically based on actual round trip time measurements. This is the approach taken by the Snoop scheme which uses a mechanism similar to TCP to track round trip delay [27]. Snoop needs adaptive timers due to contention from non TCP traffic and also because it tracks each TCP connection separately.

To avoid the complexity of dynamic timer adaptation, we experimented in our simulations with two heuristics for adjusting the timeout values, based on those appropriate for single service operation: either multiplying the base timeouts by the number of services, or multiplying them by the inverse of the fraction of the link allocated to the service. For example, consider a link with two services, with the first service allocated $1/3$ of the bandwidth. With the first heuristic, it should double its timeouts, while with the second it should triple its timeouts. The first heuristic is based on the idea that an acknowledgment may have to wait for frames from all the other services to be transmitted before its time comes, hence we should add $n - 1$ round trip

times to the timeout value to account for the other $n - 1$ services. The second heuristic is based on the idea that when the link is loaded, the effective transmission rate for the service is equal to the nominal bandwidth multiplied by the fraction allocated to the service, hence the timeouts should be correspondingly inflated by the inverse of this fraction. Although the second heuristic is probably more accurate, in those of our tests in Chapter 7 where there was a difference between the two (if n services are allocated $1/n$ of the bandwidth each, both heuristics give the same results), we found the first heuristic to work equally well.

The first heuristic is preferable because it allows the timeouts to be adapted only when the administrative module is used to add or remove services, which should be a rare event. With the second heuristic on the other hand, timeouts should be adapted every time the service rates change, which may be quite frequent when they are estimated dynamically by the classifier. In both cases, timers can be adapted in a service independent manner by adding to each service module a call that takes as a parameter a multiplicative factor, which would be either the number of services or the inverse of the service rate. Modules that use timeouts can multiply their fixed base values by this amount, unaware of the heuristic used. Either the administrative module or the classifier would issue these calls, depending on the heuristic used.

6.2.3 Scheduler

The frame scheduler used by our multi service link layer is assigned the task of sharing the physical link between all services. It must strictly enforce the service rates set by a higher layer directly or indirectly, through measurements by the classifier, when demand for the link exceeds its capacity. When the link is unloaded, services should be allowed to use more than their allocated shares to drain their buffers and queues so as not to underutilize the limited wireless bandwidth. The scheduler should allow both the number of services and their service rates to be changed dynamically, to enable services to be added and removed at any time and their rates to be readjusted, by either the administrative module or the classifier. Finally, it should be efficient in terms of time and space requirements, since it is operating at a very low level. In our design the scheduling module is independent of the actual services present on the link, being unaware of their semantics and overhead requirements. Hence the scheduler is a reusable component.

Due to the very generic nature of our requirements, many different schemes may be used. We concentrated on *work conserving* schemes, which never leave the link idle when there

are frames to send. In contrast, *non work conserving* schedulers may leave the link unused when there are frames that could be transmitted but they are not *eligible* for transmission [73]. Non work conserving schemes are useful when end-to-end delay and jitter bounds need to be provided. At the link layer we are unaware of such end-to-end requirements for each frame, so work conserving schemes are preferable since they also fully utilize the limited bandwidth of wireless links. A class of schemes that fit our requirements very well are the *fair queueing* schemes, where each service i is characterized by a positive real number r_i, its *rate*. In an ideal fair queueing scheme, at any time t the actual service rate for service i is $\frac{r_i}{\sum_{j \in A(t)} r_j}$ where $A(t)$ is the set of services that have data to send at time t. Hence, r_i is the relative fraction of bandwidth allocated to service i. When some services do not have frames to transmit, their bandwidth is allocated among the rest of the services in proportion to their r_i, since the denominator $\sum_{j \in A(t)}$ only includes active services. Note that $\sum r_i$ can be any positive number. Hence, this scheme can satisfy our requirements if we just set the r_i for each service i equal to the desired fraction of bandwidth allocated to it, in which case $\sum r_i = 1$.

This ideal scheme cannot be implemented in practice as it requires all active services to transmit simultaneously infinitely divisible pieces of data. A practical scheme can only transmit one full frame after another. There are many approximations to an ideal fair queueing scheme, for example *packet generalized processor sharing* (PGPS) [77] and *weighted fair queueing* (WFQ) [78]. In these schemes, after one frame transmission completes, the scheduler transmits next the frame that would first complete transmission in an ideal fair queueing system. A refinement, called *worst-case fair weighted fair queueing* (WF^{2}Q) [79], uses the same rule but only considers frames that would have started transmission in the ideal fair queueing system. All these schemes are based on the concept of the *virtual time* of a frame: this is the time when the frame would complete transmission in the ideal fair queueing system. Frames are transmitted based on their virtual time. To determine virtual time however, a reference ideal fair queueing system must be maintained, which is computationally expensive.

Since we needed a very efficient scheduler, we decided to use a more crude approximation to ideal fair queueing, *self-clocked fair queueing* (SCFQ) [80]. In SCFQ the current virtual time is approximated by the virtual time of the frame currently being transmitted. This means that instead of continuously updating a reference server, we only need to update the virtual time whenever a frame starts transmission. In an SCFQ system, the scheduler maintains one queue

per service, transmitting frames from each queue in order. When the current frame completes transmission, the scheduler examines the frames at the head of each queue and selects for transmission the frame with the smallest virtual time stamp. The frame is removed from the queue, its virtual time stamp becomes the system's virtual time, and the frame is transmitted. Although SCFQ is equally fair to other schemes, its delay and jitter bounds are more relaxed than WFQ and WF^{2}Q. The link layer however is not aware of end-to-end jitter and delay requirements, hence fine grained jitter control is rather inappropriate at this level.

When the k-th frame for service i arrives at time t, it is added to the end of queue i. Its virtual time stamp, $V_i(k)$, is calculated from the current system virtual time, $V(t)$, the virtual time stamp of the previous frame in queue i, $V_i(k-1)$, the service rate, r_i, and the frame's length, $L(k)$, as follows:

$$V_i(k) = \frac{L(k)}{r_i} + \max(V_i(k-1), V(t))$$

Note that link bandwidth is not used for the calculation, only the relative bandwidth shares matter. The first term is the size of the packet inflated by the fraction of the link allocated to the service: the smaller the fraction, the larger the term. This depicts the virtual time needed to transmit the frame. To determine its virtual finish time, this has to be added to the virtual finish time of the previous frame. If that frame was transmitted in the past, the system virtual time is used instead. In an SCFQ scheme then, the only calculations are setting the virtual time stamp for each frame upon its arrival, and choosing the smallest virtual time among all service queue heads when a frame completes transmission.

For our purposes, we decided to incorporate the SCFQ scheme in the frame scheduler as shown in Figure 6.3. Frames arriving from service i are passed to the *time stamper*, which first sets a frame header field showing which service the frame belongs to, so that its peer service module may be later identified at the receiver. The time stamper uses the value stored at the *virtual time* variable as $V(t)$ and the virtual time of the last frame in queue i for $V_i(k-1)$, which may be found in constant time by keeping a pointer to each queue tail. Note that if queue i contains any frames, $V_i(k-1) \geq V(t)$, since the frame transmitted must have at least the same virtual time as any waiting frame, hence for non empty queues we only need $V_i(k-1)$, while for empty queues we only need $V(t)$ for the calculations. The frame header is used to find the frame size $L(k)$, which may have been inflated by service specific headers. Finally, the *rate table*,

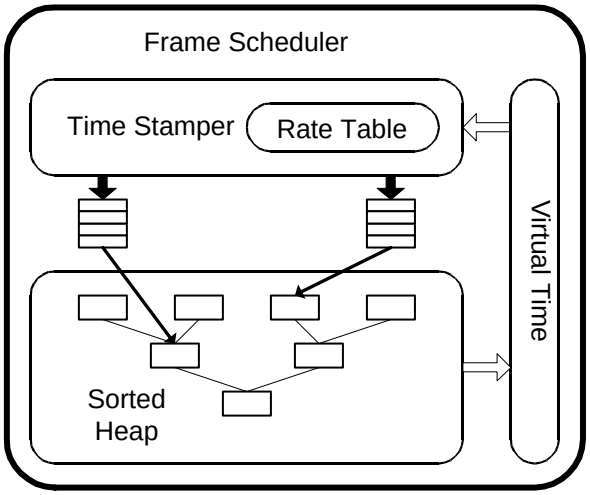

Figure 6.3: Multi service link layer scheduler

set by the administrative module or the classifier, is consulted for r_i. The virtual time stamp is calculated as shown above and stored in a temporary field in the frame header, and the frame is added at the tail of queue i. Note that these queues are *separate* from the internal queues and buffers used by service implementations: once frames are passed to the scheduler and stored in the queue, they are out of the reach of their services. The purpose of these queues is to buffer frames from one service while other services are transmitting. The cost of the timestamping process is a constant number of lookups and arithmetic functions, performed exactly once per frame. Note that for fixed frame sizes $L(k)$ is constant, so $\frac{L(k)}{r_i}$ may be calculated only once every time r_i is set to a new value.

The second function of the scheduler is selecting the next frame to transmit after each frame completion. This requires locating the queue whose first frame has the smallest virtual time in the system, dequeueing this frame, and setting the value of the *virtual time* variable equal to the frame's virtual time before transmitting it. Note that only the first frame of each non empty queue must be considered, hence we can speed up the selection process by keeping all such frames in a *sorted heap*, as shown in Figure 6.3. The heap is a binary tree, usually implemented as an array, containing n pointers, one for each of n services. The l first pointers show the l active queues in the system, i.e. those containing some frames, and the rest show the inactive ones. These l pointers are arranged so that the frames they point to conform to the order

imposed by the heap structure: the frame pointed at by any node of the tree has a smaller virtual time than the frames pointed at by its descendants. At any given time, the first element of the array is the top of the heap (the root of the tree), and points at the next frame to transmit, hence selection of the next frame for transmission has a constant cost given a sorted heap.

There are two events that cause the heap to be reordered. When a frame is selected for transmission, its queue is removed from the heap if it becomes empty, else the new head of the queue starts representing it. In the former case the last element of the heap is moved to the top, in the latter the new head remains at the top. In both cases, the heap is reordered in $lg(n)$ steps by shifting the new top towards the bottom of the heap, using the virtual time stamps of the head frames for comparisons. If an empty queue receives a frame, it is inserted at the bottom of the heap, and shifts upwards until it reaches its proper position. This also requires $lg(n)$ steps. Each frame in the system may thus cause the heap to be reordered only once or twice: if it arrives at an empty queue, the queue needs to be inserted in the heap, while after it is selected for transmission the heap must always be reorganized. Hence the total cost of maintaining the heap is bounded by $2 * \lg(n)$ comparisons and pointer exchanges for n services. At the link layer we should only support a small amount of services to avoid complexity, so this value can be seen as a constant, for example with $n = 8$ services at the link $\lg(n) = 3$. Since time stamping also takes a constant time per frame, we conclude that in practice frame scheduling requires a small constant number of operations per frame. The space requirements are also small, n heap entries and a small amount of buffers for the per service queues. Recall that the classifier only allows the IP layer to inject packets at the link layer at the nominal rate supported by the link, hence preventing these queues from overflowing.

The scheduler is completely independent from the schemes implemented and their semantics, it only sees frames coming in from the services with possibly variable sizes. The number of services and service rates are set by the administrative module and maybe the classifier. Adding or removing a service simply requires changing n and creating or destroying a service queue and a heap element. The service numbers used do not have any global significance, they only identify one of the services present on the link. This means that the link layer does not need to be aware of any global service identification scheme and that a small header field can be used to identify the n locally present services. The only link dependent aspect of the scheduler is that it must interface with the existing link layer in order to transmit frames, which means using a

link specific send call and handling frame completion interrupts. This is a task performed by the IP layer in single service schemes, so the required interfacing code for any given link and operating system may be copied from the IP layer when it is modified to communicate with our multi service link layer.

6.2.4 Demultiplexer and Multiplexer

The frame demultiplexer and the packet multiplexer are the simplest components of the multi service link layer. Their purpose is to provide a uniform input and output interface to each service on the receive part of the module. The frame demultiplexer receives frames from the existing link layer, intercepting the calls or interrupts that were originally intended for the IP layer. The frames are already stripped of existing link layer header and trailers, but they contain a service number field added by the frame scheduler. The demultiplexer strips this field and passes the frame to the appropriate service for receive processing. The services may perform any operations desired based on the received frames, including sending positive or negative acknowledgments via the frame scheduler. When they want to release packets to higher layers, they pass them to the packet multiplexer, which in turn passes them to the IP layer using a call or interrupt depending on the implementation. All services thus take their input from the demultiplexer and pass their output to the multiplexer in the receive direction, regardless of their implementation. As in the send direction, a common set of input and output calls is used by the services, hence the demultiplexer and multiplexer are service independent. The code needed for the link and operating system dependent interface with adjacent layers must be copied from the interface between existing layers, but otherwise these modules are link independent.

6.3 Evaluation

In the following paragraphs we evaluate our multi service link layer design, presented in Section 6.2, with respect to the requirements formulated in Section 6.1. We also discuss how our proposal differs from related research on the field. Let us first examine the requirements that could not be met by traditional single service link layers.

Isolation: Our design isolates services from each other by using a common set of input and output mechanisms for communication with service independent modules that in turn in-

terface with adjacent layers, providing a *virtual private* path between peer service modules. The frame scheduler allows each service to introduce arbitrary amounts of overhead. Thus, existing schemes can be used nearly unmodified.

Fairness: The fair queueing frame scheduler chosen protects the bandwidth allocated to each service from the error recovery overhead introduced by other services. When the link is loaded, services are rate limited to their allocated shares. When the link is unloaded, active services share the bandwidth unused by inactive services in proportion to their service rates, never leaving the link idle when there are frames to send.

Extensibility: The link independent modules are unaware of the semantics of each service, using standardized interfaces to communicate with them. Arbitrary services may be added and removed by simply notifying those modules of the current number of services. Service rates may be automatically measured by the classifier or externally set. Only an external administrative module is aware of the semantics and the corresponding mapping between higher layer requirements and available services.

Hence, our *multi service link layer* design can satisfy the multiprotocol and extensibility requirements that could not be met by single service schemes, by seamlessly integrating diverse link layer mechanisms in one module that simultaneously supports multiple higher layer requirements. In addition, this multi service module can still satisfy all the requirements already met by single service link layers.

Location independence: Any link can be used in a symmetric manner, since each service is essentially identical to its single service counterpart. Each wireless link in a path may transparently use our multi service scheme without affecting the rest.

Adherence to layering: Despite the provision of multiple services, individual link layer services are unaware of higher layer semantics. A classifier maps packets to services based on a hashing function and lookup table created by an external administrative module.

Easy deployment: Our scheme can be independently deployed at individual wireless links. The services offered are customized to each link and are not globally standardized. Service and link independent components can be reused and services are based on existing schemes.

Resource efficiency: Memory requirements are minimal and scale linearly with the number of services, while processing scales logarithmically. For small numbers of services, the processing requirements per frame are a small constant number of operations.

We conclude that our scheme satisfies all our requirements and is therefore a *universal* solution for enhancing Internet performance over wireless links. With a modest programming effort, existing link layer schemes may be customized to a particular wireless link and integrated with the service independent components of our design, to produce a link layer module that transparently optimizes the performance of multiple types of higher layer traffic. In particular, all of the schemes that we tested for TCP and UDP applications in Chapters 4 and 5, respectively, can be combined in a multi service module so as to optimize the performance of both types of application simultaneously, as shown in Chapter 7.

6.3.1 Related Research

The observation that different higher layer requirements are best supported by different wireless link layer mechanisms is not new, although our work is the first to examine this issue experimentally to such an extent. Most related work has concentrated on allowing connection oriented services that provide *quality of service* (QoS) guarantees in wired networks to also do so over wireless links. For *Asynchronous Transfer Mode* (ATM) networks, a QoS API has been proposed that allows the parameters of a connection to be renegotiated as link conditions change due to handoffs [81]. For cellular telephony, the GSM *general packet radio service* (GPRS) supports different QoS levels per connection [82]. Another scheme for ATM based wireless networks allows each connection to choose between a real time and a non real time link layer error recovery mechanism [83]. In all these schemes, each connection is maintained separately, requiring its own link layer service instance. An existing packet scheduler must be extended to explicitly take into account the error recovery overhead generated by each available service. In contrast, our scheme uses a single service instance for all applications having the same requirements and only performs scheduling so as to protect services from each other, leaving to higher layers the task of setting scheduling priorities, if needed. As a result, our scheme is simpler to implement and easier to extend. In addition, it is compatible with the existing Internet where packet scheduling is not provided, in contrast to ATM networks. We further discuss in

Chapter 8 how emerging Internet oriented QoS provisioning approaches interact with our link layer scheme.

The proposal closest to our work also started from the viewpoint of bridging ATM and wireless, arguing that next generation cellular systems could use different link layer schemes for each ATM connection depending on its QoS requirements [84]. This evolved into a scheme for IP oriented QoS in which two services were provided, a reliable ARQ based scheme (a cross between our GBN and SR schemes) and an unreliable scheme that basically exposes the raw link [85]. Higher layer traffic is allocated to one of these services based on its *delay* requirements. This scheme, proposed virtually at the same time as ours (see references [72] and [85]), is very limited compared to our approach. It globally defines two services and their implementations, despite having tested only a single scheme with UDP applications over a specific wireless link, and only with the purpose of measuring delay. Our multi service link layer scheme leaves open the number of services to be offered, their implementation and the mapping from higher layer traffic to link layer services, to allow for extensions as future needs arise. We have implemented and tested a large number of diverse schemes with multiple applications and wireless links, to offer input for this process. While the two service proposal always uses the raw link service for delay sensitive UDP applications, we are using enhancement schemes for both TCP and UDP applications, combining the two in the measurements in Chapter 7. Although our scheme is much more flexible and extensible, it is also backed by comprehensive measurements of different QoS parameters (loss, delay, throughput), under an extensive set of wireless links, topologies and applications.

Chapter 7

Multi Service Link Layer Performance

This chapter describes the simulator implementation of our multi service link layer scheme, presents detailed simulation results and analyzes their implications. Section 7.1 shows how we implemented in the simulator the multi service link layer design presented in Section 6.2. Section 7.2 describes the applications and link layer enhancement schemes tested. Section 7.3 presents and analyzes the results of our multi service simulations. Section 7.4 summarizes our conclusions.

7.1 Multi Service Link Layer Simulator

We extensively modified the ns-2 simulator to support nearly all of the functionality described in Section 6.2. The internal design and data flow of the simulated multi service link layer module is nearly identical to that depicted in Figure 6.1. Packets received from the IP layer for transmission are passed to a classifier. The classifier blocks the IP module for the nominal duration of the packet's transmission, thus preventing IP queues from transmitting faster than the link speed. The classifier selects one of the available schemes based on the *differentiated services* (DS) field of the IP header [86], by directly mapping its value to a service number. The DS field may be set either by the application or by intermediate routers to indicate the *quality of service* (QoS) required for each packet. Although we set this field directly at the application level for all of our simulations, in general a local administrative module should either set this field to directly correspond to one of the available services or provide an appropriate table mapping global DS field values to local service numbers for use by the classifier. We defer further discussion of the

DS field and its use to Chapter 8 where we discuss the *differentiated services* framework [87] for QoS provision on the Internet and its relation to our multi service link layer.

The classifier passes the packet to one of the services, using a service independent call. Any number and type of services may be used at each link, and services may be added or removed at any time. The selected service may perform any processing required, eventually passing one or more frames to the frame scheduler. This is a *self clocked fair queueing* (SCFQ) [80] scheduler identical to the one shown in Figure 6.3. The scheduler marks the frame with its service number, adds a virtual time stamp to it and places it at the tail of its service queue. The service rate table used by the scheduler is set by an administrative module and can be modified at any time. Whenever the link becomes available, the scheduler selects the frame with the earliest virtual time stamp, using a heap structure to keep the service queues sorted at all times. Frames that are not lost in transit are received by a frame demultiplexer at the other end of the link, which strips their service number and passes them to the appropriate service. The selected service performs any processing required, possibly transmitting its own frames in reply, and eventually releases one or more packets to a multiplexer that passes them to the IP layer for further processing. All services use a common set of calls to send/receive data to/from the other multi service link layer modules in order to ease integration and allow extensibility.

When only a single service is present or active, the multi service scheme operates as if it was a simple single service link layer: since the scheduler is work conserving and it sends frames from each queue in a *first-in first-out* (FIFO) fashion, frames are sent in the same sequence and at the same times as without the scheduler. The service rate used for the active service is irrelevant in this case. Actually, we used our multi service module with a single service for all the tests presented in Chapters 4 and 5. The only difference to a single service link layer was the requirement for a service number in the frame header, but as this was added to all services, it did not influence their relative performance. Note that both the services and the link independent components are reusable, and we actually used the same code for tests with all the simulated links. The services were implemented by inheriting from a base service object that contained the common input and output calls to communicate with other multi service link layer components, but otherwise provided no link enhancements. This was the *Default* service shown in our results, which offered access to the raw link.

Link Type	TCP Rate	TCP Bandwidth	UDP Rate	UDP Bandwidth
Cellular	0.33	4.8 Kbps	0.66	9.6 Kbps
PCS	0.50	32 Kbps	0.50	32 Kbps
WLAN	0.50	1 Mbps	0.50	1 Mbps

Table 7.1: Service rates and bandwidths for multi service tests

7.2 Simulated Applications and Link Layer Schemes

In order to evaluate the performance of our multi service link layer scheme, we simulated a mix of TCP and UDP based applications executing simultaneously over each wireless link, using different link layer enhancement schemes for TCP and UDP. We employed the same application models that were presented and tested in Chapters 4 and 5.

File Transfer: A client receiving a large amount of data from a server using TCP, without sending any feedback. Described in Section 4.1.1.

World Wide Web Browsing: A WWW client using TCP to access a WWW server. Data transfer is bidirectional but mostly flows towards the client. Described in Section 4.1.2.

Real Time Conferencing: A conference participant sending voice and maybe video to a receiver using UDP, with silence suppression. Described in Section 5.1.1.

Since we found out in Chapter 4 that both file transfer and WWW browsing performance can be optimized using the same link layer schemes, we decided to use the same service for both, and a separate service for UDP based real time conferencing with its inherently different requirements, as discussed in Chapter 5. Each application sets the DS field in all of its packets to allow the multi service link layer to select either the TCP or the UDP oriented service. To enable comparisons between the single and multi service tests, we decided to use the same bandwidth parameters for the real time conferencing application, as shown in Table 5.1. Since this application is very delay sensitive, we allocated for each link a fraction of the available bandwidth that was sufficient to serve its *peak* bandwidth requirements. Table 7.1 shows the service rates (or, equivalently, bandwidth fractions) allocated to the TCP and UDP oriented services, and the resulting bandwidth shares. Note that the actual bandwidth requirements of the UDP based application are slightly higher than its nominal transmission rate due to error recovery

overhead. Both TCP based applications share the remaining bandwidth using TCP congestion control mechanisms to avoid overloading the link. The rates shown are only enforced when there is contention for link resources, hence when the real time conferencing source is inactive, all of the bandwidth is available for TCP traffic. Similarly, UDP error recovery overhead during periods when the real time conferencing source is active can be transmitted in excess of its allocated bandwidth if TCP sources have no data to send. We used the same performance metrics for each application as in Chapters 4 and 5.

For our multi service tests we used the best link layer schemes from each family of mechanisms for TCP and UDP based applications, as determined by our single application tests.

SR/M2: A selective repeat scheme supporting multiple negative acknowledgments and full recovery for TCP. Described in Section 4.2.1

RLP: A negative acknowledgment only selective repeat scheme supporting limited recovery for TCP, with up to 3 retransmissions. Described in Section 4.2.2

Snoop: A TCP aware scheme supporting retransmissions from the base station and TCP acknowledgment suppression. Described in Section 4.2.3

RLP/OOS: A variant of RLP that releases received frames out of sequence to real time UDP applications, with up to 1 retransmission. Described in Section 5.2.2

For each test, we used either SR/M2, RLP or Snoop for both TCP applications and RLP/OOS for the UDP application, and compared their performance against using the Default (no enhancements) scheme for both TCP and UDP applications. While Snoop dynamically estimates its timeout values based on local round trip time measurements, the other schemes had to explicitly relax their timeout values to account for the increased contention on the wireless link, as discussed in Section 6.2. For the PCS and WLAN links, both heuristics presented in Section 6.2 for inflating timer settings suggest doubling the timers (2 services with $1/2$ of the bandwidth allocated to each). For the Cellular link, the two heuristics differ, for example for the TCP service we may either double the timers (2 services) or triple them ($1/3$ of the bandwidth). We found that using the number of services produced similar results to using the link shares, so we used the former heuristic as it is simpler and does not require recomputing timeout values every time the service rates are modified.

7.3 Simulation Results

In the following paragraphs we present and analyze application performance results from simulations of simultaneous file transfer, WWW browsing and real time conferencing using ns-2. We are mainly interested in examining how the multi service link layer scheme affects performance compared to single application tests. We present the metrics for each application separately to see how each type of link behaves with different mixes of TCP and UDP oriented services. Although the link layer scheduler protects each service from the error recovery overhead introduced by the rest, the TCP oriented service supported both file transfer and WWW browsing. As a result the bandwidth available to this service had to be shared by the two applications using TCP congestion control mechanisms. This means that their performance metrics may depend on each other, hence they should also be examined in a comparative manner. Each test was repeated 30 times with different random number generator seeds. The metrics presented reflect averages from all these tests, while the error bars shown depict mean values plus and minus one standard deviation. We started all applications at the same time and ended the simulation when the file transfer application completed a fixed size transfer. The amount of data sent for each wireless link was the same as shown in Table 4.1. We used the same TCP and UDP simulator objects and parameters as in Chapters 4 and 5 to ease comparisons (for TCP this meant Reno, 500 ms granularity timers and one acknowledgment per packet) as well as the same wireless link parameters, including frame and packet sizes.

The application peers were located on each topology in the same manner as in previous tests. In one wireless link topologies, the main traffic direction was from the wired host towards the wireless host, meaning that the wired host was the data source for file transfer, the server for WWW browsing and the speaker for real time conferencing. File transfer also generated traffic in the reverse direction in the form of TCP acknowledgments, while WWW browsing generated data and acknowledgments in both directions due to its interactive nature. The real time conferencing application was only simulated in one direction, hence no feedback was generated by the receiver. In two wireless link topologies, one of the wireless hosts was chosen arbitrarily to play the same role as the wired host in one link tests for all applications, i.e. the same wireless host was the data source, WWW server or speaker. As a result, in each topology all applications shared the same end-to-end path.

7.3.1 File Transfer Performance

Figure 7.1 shows the average file transfer throughput achieved on multi service tests in the Cellular/LAN1 scenario, i.e. a path consisting of one Cellular and one wired LAN link. Each line shows performance under a different combination of schemes for TCP and UDP traffic at various error rates. The first scheme mentioned in each figure is used for TCP and the second for UDP. The *Default* line reflects tests where both TCP and UDP employed the Default service, but each type of traffic used an *independent* service module, with the two services sharing the link following the rate allocations shown in Table 7.1. In order to compare the performance of the multi service scenario to previous results, we should bear in mind that both file transfer and WWW browsing share the bandwidth available to their service. Although the service rate for TCP is only $1/3$, which implies 4.8 Kbps of available bandwidth, the real time conferencing source only transmits at its peak rate when active, so the average available bandwidth for TCP is actually around 10.3 Kbps (see Table 5.1). However, due to the conservative nature of TCP congestion control mechanisms, there is some delay between the UDP application releasing its bandwidth (when the source becomes inactive) and TCP seizing it.

Interestingly, using SR/M2 or RLP for file transfer in this scenario leads to *worse* throughput results than with the Default scheme at low error rates, a rather unexpected result given the superiority of these schemes over Default in single application tests. Although use of RLP/OOS for UDP as opposed to Default means that there is slightly more contention on the link due to retransmissions, the real explanation for this drop in throughput is the corresponding increase in WWW browsing throughput in this scenario against the Default scheme, shown in Figure 7.7. WWW browsing consists of small transfers that often do not have enough data to transmit to trigger TCP error recovery after a loss by generating sufficient duplicate acknowledgments. In contrast, file transfer always has data to transmit until the end of the simulation. As a result, although both file transfer and WWW browsing suffer from losses when using the Default scheme, WWW browsing relies more on slower TCP timeouts for recovery, hence leaving more bandwidth available to file transfers. With SR/M2 and RLP on the other hand, WWW browsing manages to get a better share of the link when competing against file transfer, leading to a drop in file transfer throughput. While WWW browsing can only improve its unidirectional throughput metric by a small amount since as a request and reply application is has periods of inactivity in each direction, even a small increase in traffic causes TCP connections to back off

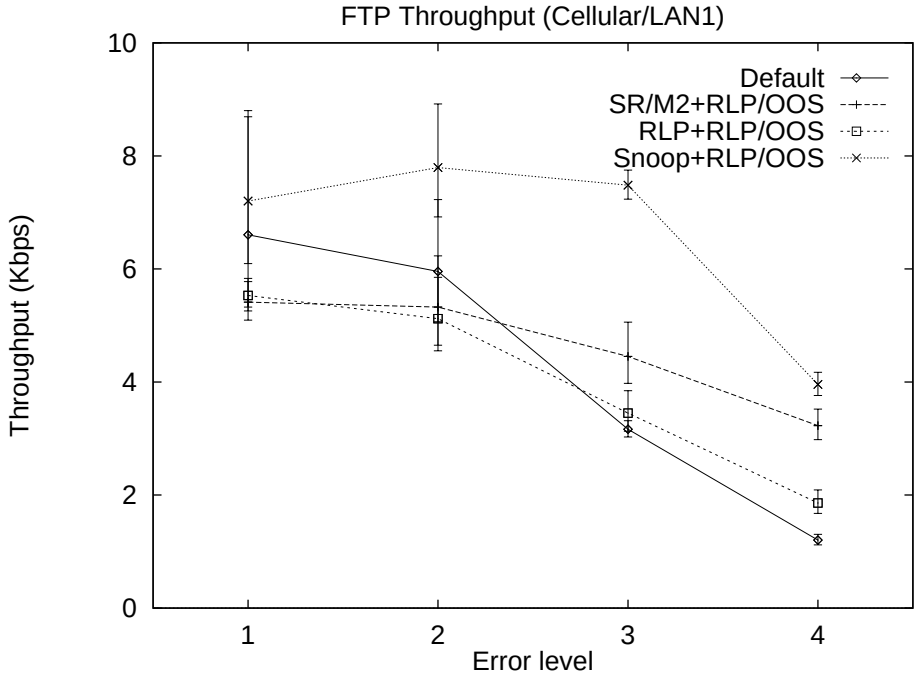

Figure 7.1: File transfer throughput over one multi service Cellular link

to avoid congestion. Hence, the drop in file transfer throughput is larger than the increase in WWW browsing throughput. These effects are further exaggerated by the limited Cellular link bandwidth which makes TCP congestion control more sensitive.

Given these observations, the performance results are otherwise what should be expected. RLP is slightly faster than SR/M2 at low error rates as its negative acknowledgments are more efficient, but loses its advantage very fast since SR/M2 is more robust as error rates increase and RLP starts giving up on some frames. The Default scheme also loses its advantage when higher error rates force the TCP sender to waste more time waiting for timeouts to occur. The Snoop scheme performs best for the same reasons that the Default scheme achieved higher throughput than SR/M2 and RLP, i.e. even further reduced WWW browsing performance, that is even worse than Default at low error rates (see Figure 7.7). In this case, since Snoop is ineffective for the bidirectional traffic of WWW browsing, but effective for the unidirectional file transfer, it manages to improve file transfer performance over the Default scheme. This causes WWW browsing bandwidth to be further reduced as the increased file transfer bandwidth produces even more contention on the link, further delaying the already disadvantaged WWW browser.

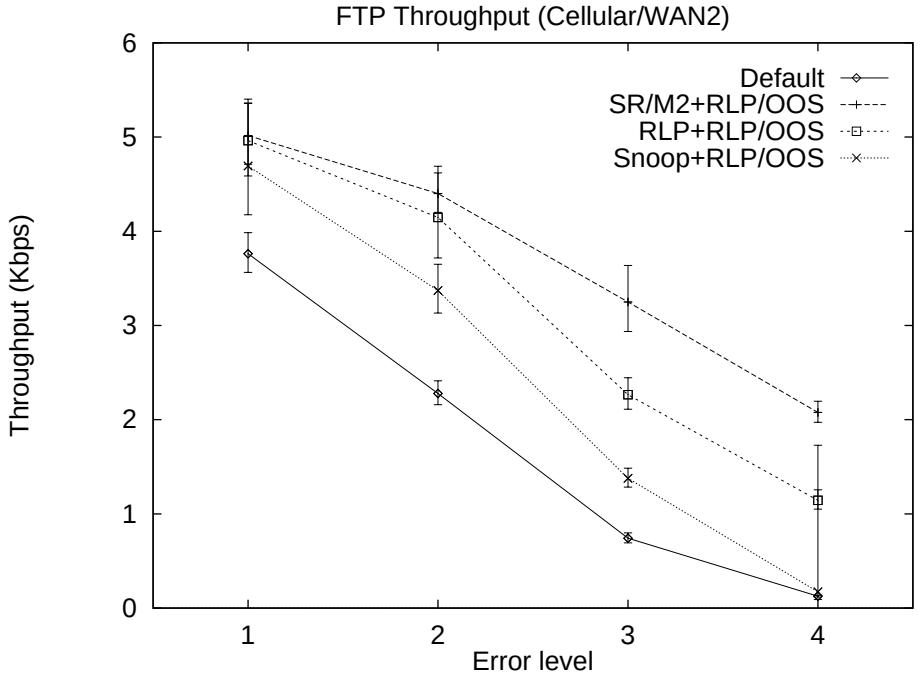

Figure 7.2: File transfer throughput over two multi service Cellular links

In the two Cellular link Cellular/WAN2 scenario, as shown in Figure 7.2 the results are closer to what would be expected based on single application tests. SR/M2 performs best, with RLP becoming an increasingly distant second as the combined error rate of two wireless links makes it give up on more frames, hence propagating more losses to TCP. Snoop loses its lead as it is not effective against the losses in the first wireless link of the path, being a base station oriented solution. Default is even worse as no error recovery is provided at all. End-to-end recovery is also slower in this case due to the extra delay introduced by the WAN wired path, further degrading performance when it has to be used (rarely for RLP, frequently for Snoop, always for Default). Compared to single service results, the gap between SR/M2 and RLP and the less effective Snoop and Default schemes is less pronounced (see Figure 4.3), again due to a corresponding improvement in WWW browsing performance in this scenario (see Figure 7.8). The shapes of the throughput lines in the single and multi service cases are nearly identical, adjusted for the differences in total available bandwidth and the exact balance between file transfer and WWW browsing performance for each TCP oriented service. In Cellular link tests, lower error rates lead to higher throughput variance, due to increased contention between

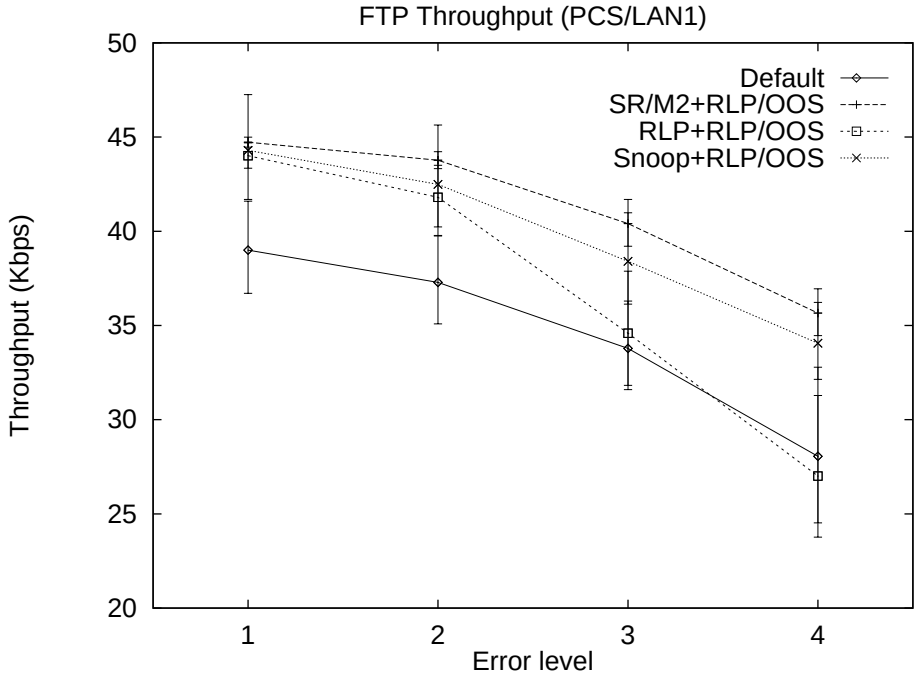

Figure 7.3: File transfer throughput over one multi service PCS link

file transfer and WWW browsing. The improvement offered by SR/M2 over the Default scheme in this scenario is in the 33-1550% range.

On the faster, but harsher in terms of errors, PCS links, the situation is very similar to that of single application tests. In this case both TCP applications share at least half of the available bandwidth, i.e. 32 Kbps, while the average available bandwidth for TCP considering the long term average transmission rate of real time conferencing increases to 50.4 Kbps (see Table 5.1). In the one wireless link Cellular/LAN1 scenario, file transfer throughput, shown in Figure 7.3, is optimized by the more robust SR/M2, with Snoop trailing by a small margin. RLP starts close to SR/M2 and Snoop but drops faster than both, eventually ending up worse than Default at the highest error level, due to its limited recovery. The shape of all throughput lines is nearly identical to their counterparts in single application tests, adjusted for reduced available bandwidth due to contention (see Figure 4.5), but Snoop and Default are relatively closer to SR/M2 and RLP. The reason is again their considerably worse performance in WWW browsing tests, as shown in Figure 7.9, which leaves more bandwidth available for file transfer. The throughput improvement offered by SR/M2 over the Default scheme in this scenario is limited

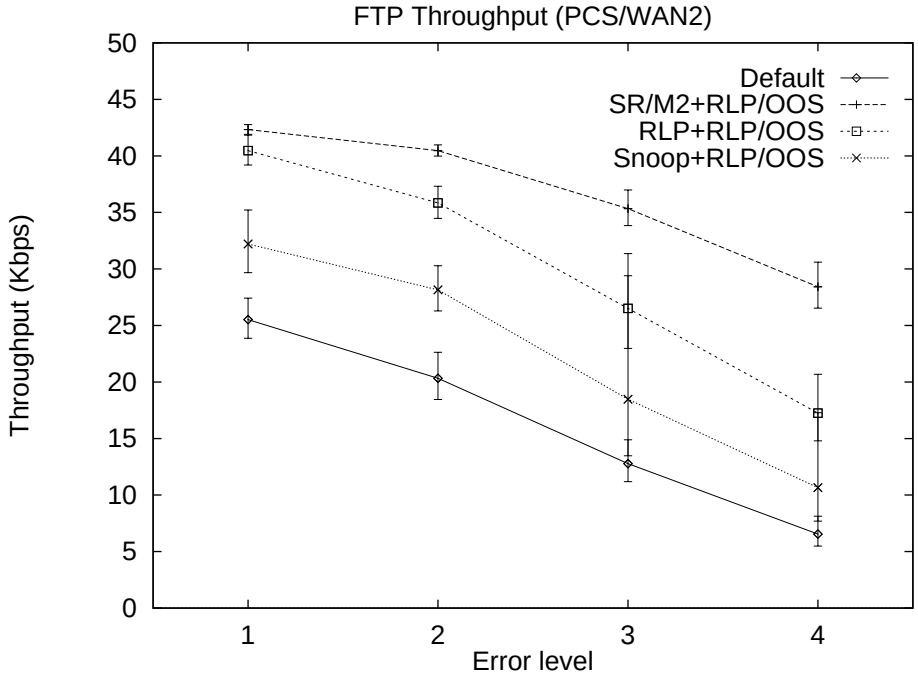

Figure 7.4: File transfer throughput over two multi service PCS links

to 15-27%. This is because the link is more fairly shared between file transfer and WWW browsing traffic with SR/M2, avoiding the bias in favor of file transfer seen with Default.

The two PCS link scenario PCS/WAN2 also mirrors single application test performance, as seen by comparing Figure 7.4 with Figure 4.6. SR/M2 offers the best performance, with RLP close at low error rates but increasingly suboptimal at higher error rates. Snoop again fails to sufficiently enhance performance due to its inability to deal with two wireless links. In this case not only the shape of the throughput lines is similar, but their relative positions are also roughly the same, despite the fact that both Snoop and Default also depict worse WWW browsing performance (see Figure 7.10). The reason is that these schemes were already problematic for this topology, even without contention from WWW browsing traffic. The throughput improvement offered by SR/M2 over the Default scheme is much more significant in this scenario, compared to its one wireless link variant, falling in the 66-335% range. The high throughput variance experienced is mainly due to the deep fading error model of PCS links.

In the faster and less error prone WLAN links, the situation is again very similar to single application tests, adjusted for the bandwidth difference. In this case the available band-

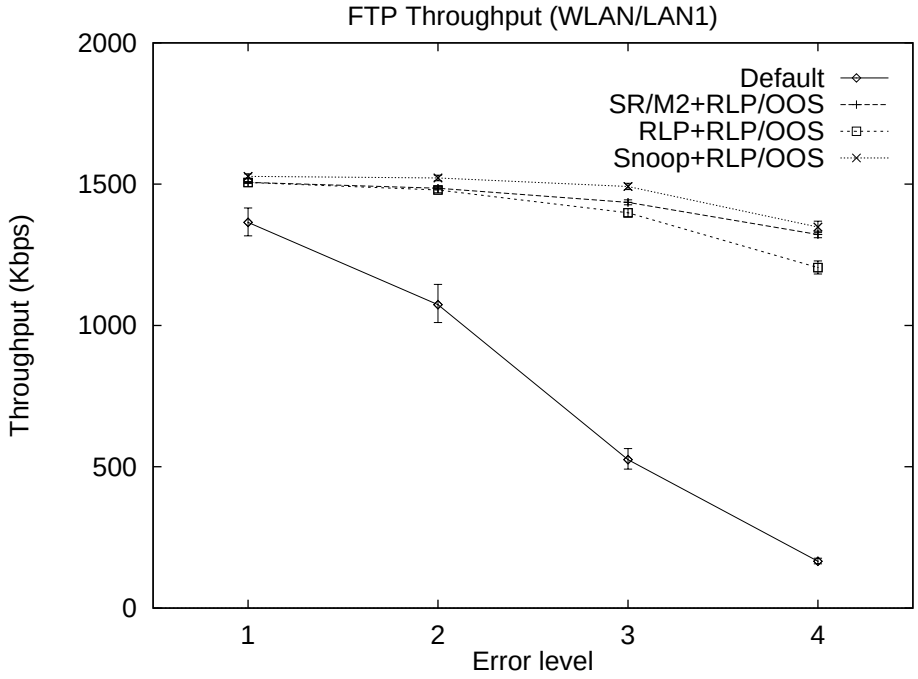

Figure 7.5: File transfer throughput over one multi service WLAN link

width for TCP applications is at least 1 Mbps, but due to pauses in the real time conferencing source the average available bandwidth for TCP increases to 1575 Kbps (see Table 5.1). In the one wireless link WLAN/LAN1 scenario, file transfer throughput, shown in Figure 7.5, reveals that Snoop has a slight edge over SR/M2 and RLP, with Default being progressively worse. Compared to the Cellular and PCS case, the higher available bandwidth means that inactivity periods are relatively more damaging to average throughput, hence the clearly inferior performance of the Default scheme. All the other schemes offer similar performance gains as in the single application tests (See Figure 4.8), but Snoop is slightly better because not only it does not improve WWW browsing throughput as much as SR/M2 and RLP, it actually offers worse performance than the Default scheme (see Figure 7.11), hence leaving more bandwidth available for file transfer. In this scenario, the improvements over the Default scheme are 11-700% for SR/M2 and 12-715% for Snoop, very close to each other.

Finally, in the two wireless link WLAN/WAN2 scenario file transfer throughput, shown in Figure 7.6, shows SR/M2 leading with RLP close behind. Snoop and Default are much worse, with Snoop offering comparatively minor performance improvements, due to its inability to re-

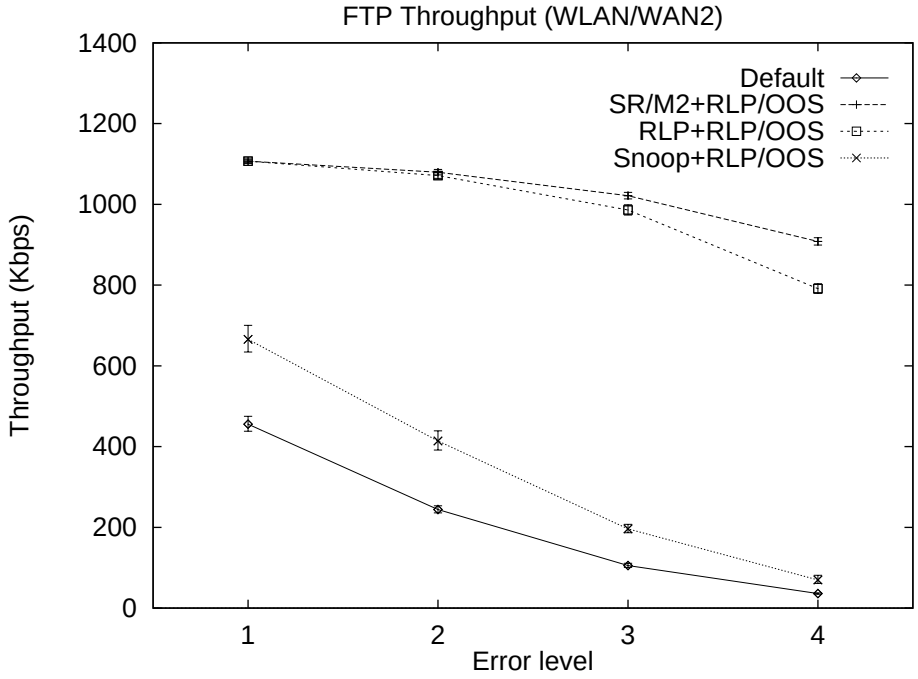

Figure 7.6: File transfer throughput over two multi service WLAN links

cover from losses on the first wireless link. The throughput lines and their relative placement are very similar to those of the single application variant of this scenario (see Figure 4.9). As in the Cellular and PCS cases, the degraded WWW browsing performance of both Snoop and Default (see Figure 7.12) does not allow them to provide better file transfer performance due to their inherent problems with the WAN2 topology. The performance improvement offered by SR/M2 over the Default scheme in this scenario are in the 143-2425% range. Throughput variance is limited due to the low error rates and high available bandwidth of WLAN links.

7.3.2 World Wide Web Browsing Performance

Based on the preceding discussion on file transfer performance, we should expect WWW browsing performance to be similar to that seen on the single application tests of Chapter 4, adjusted for reduced available bandwidth due to contention with the real time conferencing and file transfer applications. In addition, the relative performance between tested link layer schemes should change due to the different tradeoffs they offer between file transfer and WWW browsing traffic. Each figure in this section shows WWW browsing throughput in the server to

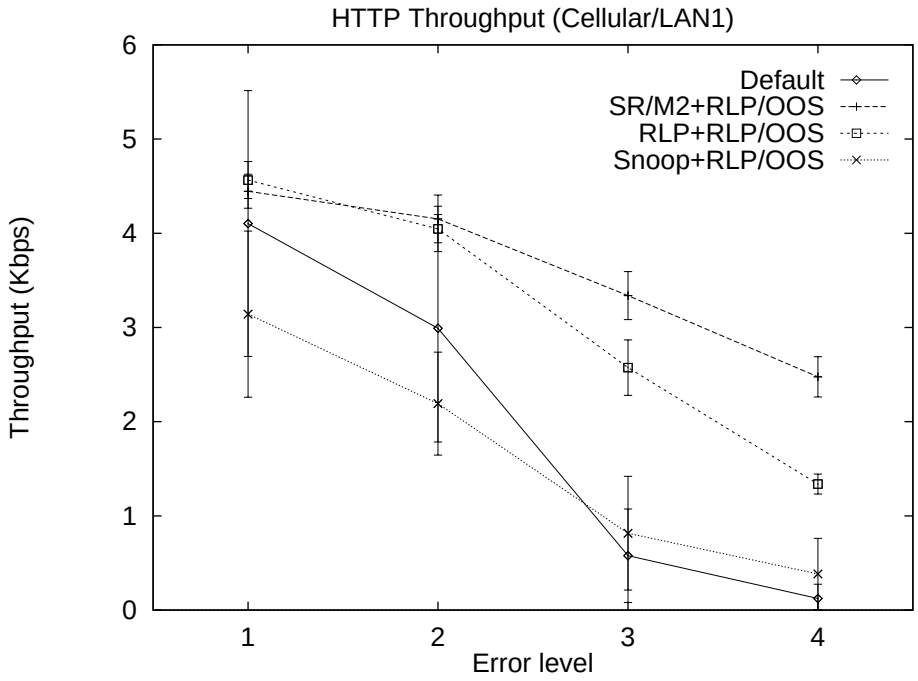

Figure 7.7: WWW browsing throughput over one multi service Cellular link

client direction. Unlike single application tests, each test repetition did not last for a constant period of time, but ended instead when the simultaneous fixed size file transfer was completed. As in single application tests, we only used the time interval until the last complete transaction ended and the amount of data transferred in that interval for throughput calculations.

Figure 7.7 shows throughput in the Cellular/LAN1 scenario, i.e. one Cellular and one wired LAN link on the path. As in the corresponding single application test (see Figure 4.11), for low error rates RLP is better than SR/M2 due to its reduced overhead and faster reaction to losses, but with increasing loss rates SR/M2 takes a clear lead due to its robustness. Interestingly, Snoop in this scenario is *worse* than the Default scheme for low to moderate error rates, offering only minor improvements at higher error rates. The reason is its bias in favor of the unidirectional file transfer application (see Figure 7.1). Since Snoop only retransmits from the base station, it only improves WWW browsing traffic in the server to client direction, with client requests in the reverse direction taking longer to complete. During these client requests, file transfer becomes more aggressive in the server to client direction. When the request completes, the server replies find the link loaded and their TCP connections back off to avoid congestion, leading to degraded

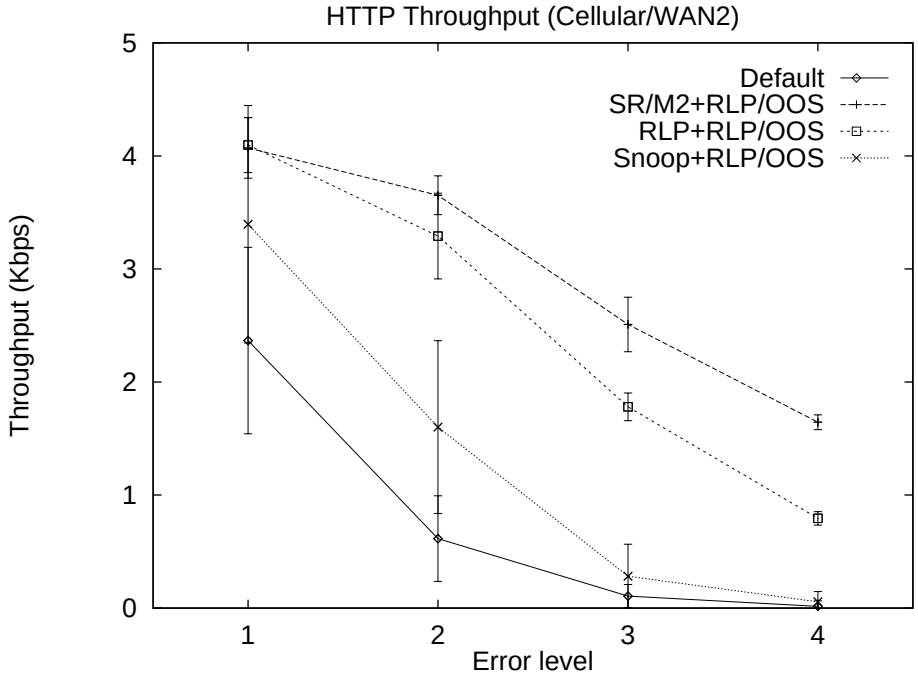

Figure 7.8: WWW browsing throughput over two multi service Cellular links

WWW browsing performance. SR/M2 and RLP work equally well in both directions, causing the available bandwidth to be shared in a more fair manner between the two TCP applications. In this scenario, WWW browsing throughput is improved compared to the Default scheme by 9-1910% with SR/M2 and by 12-985% with RLP.

With two Cellular links, in the Cellular/WAN2 scenario, the relative performance between the schemes is also similar to single application tests (compare Figure 7.8 and Figure 4.12). As in the one wireless link case, RLP has a slight edge on performance at the lowest error rate, but SR/M2 leads by a considerable margin in all other cases. The Default scheme is the worst by a wide margin, with nearly zero throughput at the highest error rate, while Snoop is in between. In this case Snoop has trouble in both the server to client and in the client to server directions since it is only effective in one direction over each wireless link. On the other hand it is more efficient than the Default scheme which besides offering no link layer enhancements suffers from longer end-to-end recovery delays due to the high delay WAN path. In this topology Snoop is also not as effective for unidirectional file transfers (see Figure 7.2), hence file transfer does not take away the available bandwidth from WWW browsing traffic. As a result, perfor-

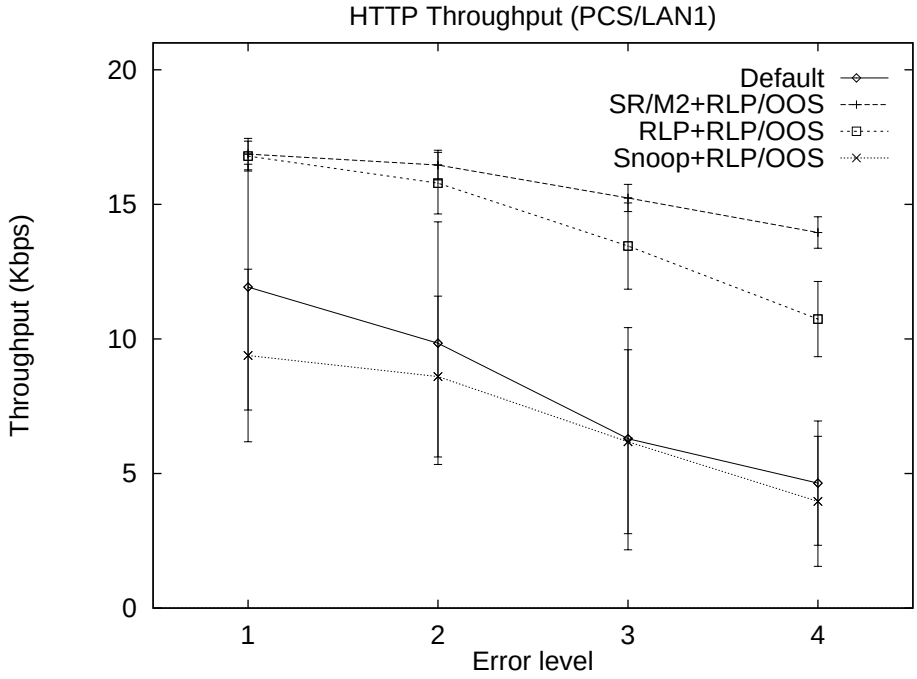

Figure 7.9: WWW browsing throughput over one multi service PCS link

mance with Snoop does not become worse than with the Default scheme as was the case with the Cellular/LAN1 scenario. Compared to the Default scheme, the performance improvements in this scenario are 72-10730% with SR/M2 and 73-5130% with RLP, considerably higher than in the single link case. In both Cellular topologies throughput variance is higher with lower error rates, as is the case for file transfers.

With PCS links the situation is similar to that with Cellular links. Figure 7.9 shows WWW browsing throughput for the one wireless link PCS/LAN1 scenario. SR/M2 consistently leads in performance, with RLP at a growing distance as error rates increase. Snoop turns out to be worse than Default in all cases, which, as in the Cellular case, is due to its inability to enhance bidirectional traffic coupled with a bias in favor of unidirectional file transfer (see Figure 7.3). In this scenario however, Snoop is worse than SR/M2 for both file transfer and WWW browsing. Performance is improved with SR/M2 compared to the Default scheme by 42-200%.

In the two wireless link PCS/WAN2 scenario, the throughput results shown in Figure 7.10 indicate similar behavior as in the corresponding Cellular link scenario. SR/M2 is the best performing scheme, followed by RLP, while Snoop and Default trail behind both. As in the

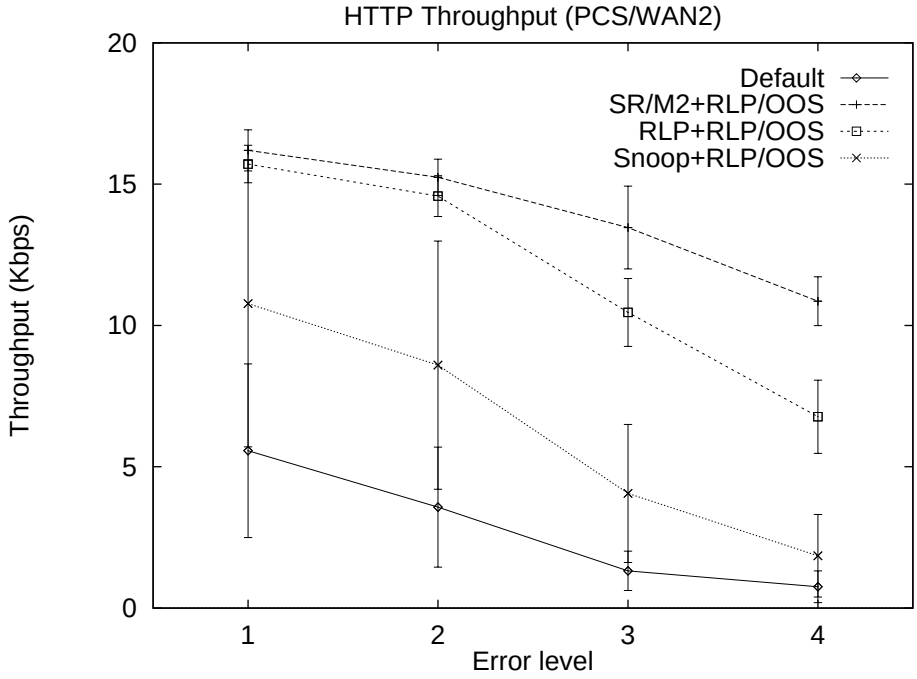

Figure 7.10: WWW browsing throughput over two multi service PCS links

Cellular case, Snoop is no longer worse than the Default scheme as it fails to improve file transfer performance sufficiently to allow it to aggressively compete for bandwidth against WWW browsing traffic (see Figure 7.4). RLP performance is worse relative to SR/M2 compared with the corresponding single application tests, with both one and two PCS links on the path (see Figures 4.13 and 4.14), indicating that RLP may be more sensitive to contention on the link than SR/M2. The performance improvement compared to the Default scheme in this scenario is 190-1340% when using SR/M2. Throughput variance is high due to the PCS link error model.

With one WLAN link in the WLAN/LAN1 scenario, WWW browsing throughput, shown in Figure 7.11, is optimized with either RLP for low error rates or with SR/M2 for high loss rates. The higher available bandwidth eliminates the sensitivity of RLP to contention that was seen in PCS tests. The Default scheme follows at a considerable distance, and Snoop is again worse than the Default scheme for low to moderate loss rates. Although this is partly due to its bias in favor of unidirectional file transfer (see Figure 7.5), WWW browsing performance with Snoop was already nearly identical to that with the Default scheme in single application tests (see Figure 4.15). The relative difference between SR/M2 and RLP in one hand and Snoop

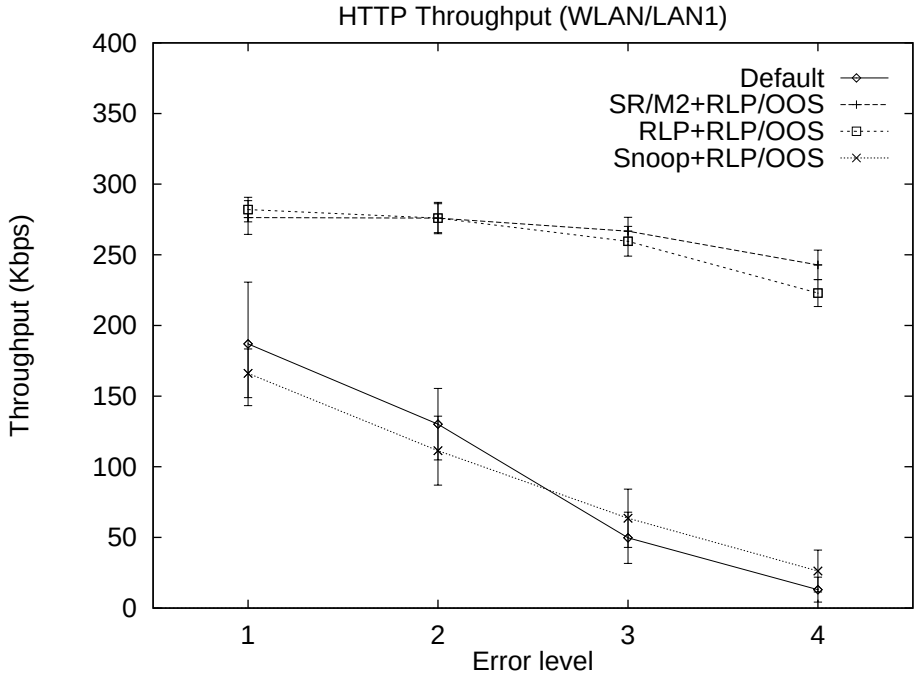

Figure 7.11: WWW browsing throughput over one multi service WLAN link

and Default on the other are larger than in Cellular and PCS links due to the increased effect of losses on average throughput with faster links: end-to-end recovery takes similar time to be triggered by a timeout in all cases due to the coarse grained TCP timers (500 ms), which means that more bandwidth is wasted when faster links are used. In this scenario, throughput improves compared to the Default scheme by 48-1760% with SR/M2 and by 51-1605% with RLP.

Finally, with two WLAN links in the WLAN/WAN2 scenario, throughput behaves similarly to the PCS case, as shown in Figure 7.12. SR/M2 and RLP are very close to each other in terms of performance, with SR/M2 leading by a small margin. The Snoop scheme in this case improves over Default as it does not take away bandwidth from WWW traffic to benefit file transfer (see Figure 7.6), but the distance between Snoop and SR/M2 is considerable. These are the expected results given the inability of Snoop to deal with multiple wireless links and bidirectional traffic and the inefficiency of end-to-end recovery over high speed links and long delay paths. These results also agree with those of the corresponding single application test (see Figure 4.16). Performance in this case improves by 153-6735% when using SR/M2, compared to the Default scheme. Similar to file transfer, throughput variance is limited with WLAN links.

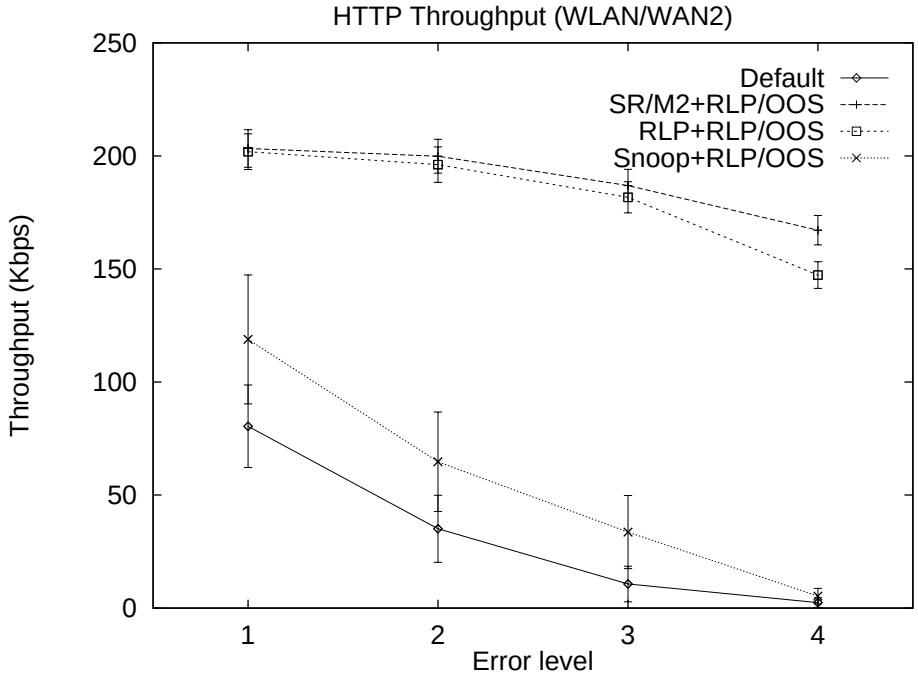

Figure 7.12: WWW browsing throughput over two multi service WLAN links

7.3.3 Real Time Conferencing Performance

In multi service tests, the real time conferencing UDP based application was started simultaneously with file transfer and WWW browsing, and its performance was measured until the end of the fixed size file transfer. We tested the UDP based application over both the Default and the RLP/OOS limited recovery ARQ scheme, with up to one retransmission per frame. In all scenarios we ensured that the scheduler used service rates for UDP traffic that allowed the real time conferencing application to transmit at its peak rate when active without experiencing congestion (see Table 7.1). As a result, despite contention from TCP traffic, the UDP based application could always get enough bandwidth to satisfy its nominal requirements. Note that when using the RLP/OOS scheme, real time conferencing bandwidth requirements were inflated due to retransmissions, but since only one retransmission was allowed per frame, the overhead generated was small enough to be absorbed by the link when the TCP service was inactive.

Regarding loss rates, performance was also the same as with the corresponding single service scenarios. Even though two services (and three applications) were sharing the link, over

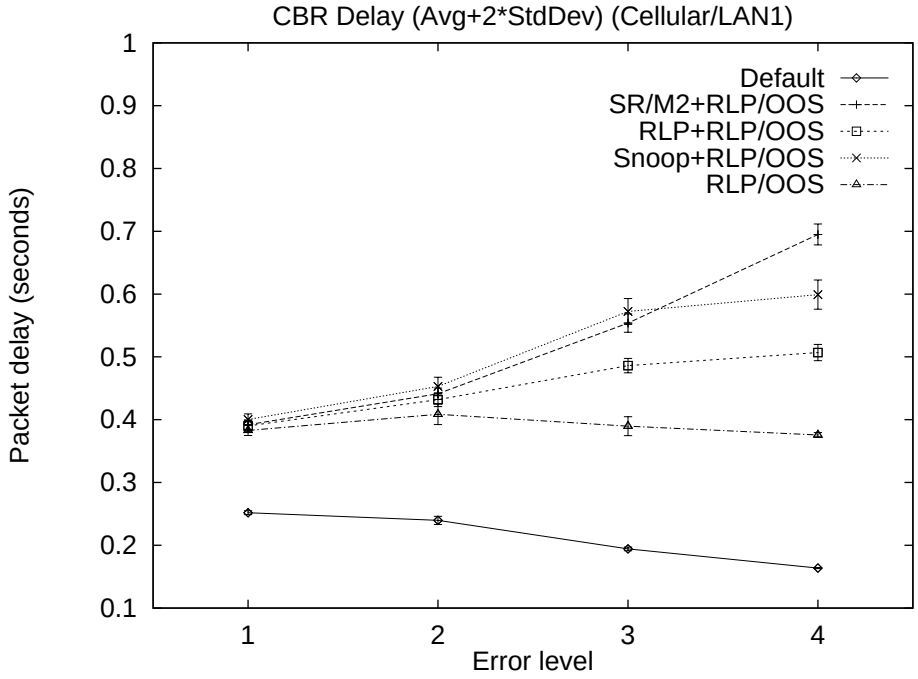

Figure 7.13: Real time conferencing delay over one multi service Cellular link

a long period of time each application experienced the same error behavior from the wireless links. Regardless of the scheme used by the TCP oriented service, the effective loss rate for the UDP oriented service only depended on whether the Default or the RLP/OOS scheme was used. In turn, these schemes performed exactly the same as in the single application tests, after we adjusted the RLP/OOS retransmission timers to account for the increased contention on the link in the same manner as for the TCP oriented RLP and SR/M2 schemes. For brevity, we omit figures showing the loss rates of the Default and RLP/OOS schemes in the multi service tests, as the corresponding figures in Chapter 5 present the exact same results.

Figure 7.13 shows average end-to-end delay plus twice its standard deviation for real time conferencing in the one Cellular link Cellular/LAN1 scenario. We present the same tests as for the TCP applications: using the Default scheme for both the TCP and UDP oriented services, and using RLP/OOS for the UDP oriented service with SR/M2, RLP and Snoop for the TCP oriented service. We also show results from tests with RLP/OOS as the UDP oriented service and TCP using the Default service. The Default line shows how much delay is due to contention with the (not enhanced) TCP applications, while the RLP/OOS line also includes the

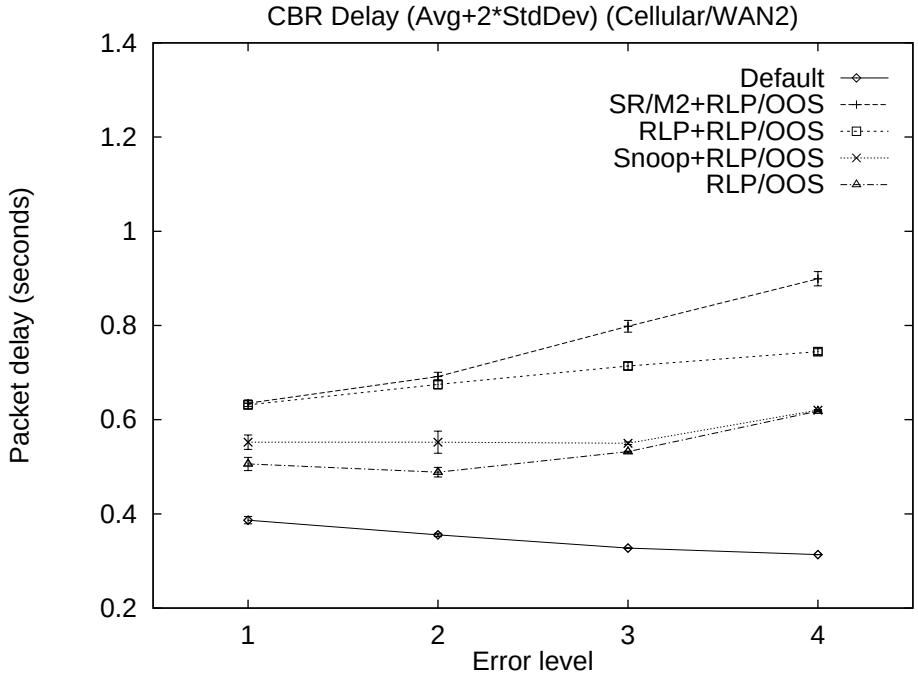

Figure 7.14: Real time conferencing delay over two multi service Cellular links

delay introduced by retransmissions of real time conferencing frames. Any additional delays in the other three lines are due to the increased contention on the link caused by retransmissions for TCP application traffic. Thus, in each figure it is made clear how much of the delay should be attributed to contention with TCP data, UDP retransmissions and TCP retransmissions.

In the single Cellular link scenario, delay with the Default scheme drops with higher loss rates as the TCP applications reduce their throughput (see Figures 7.1 and 7.7), hence also reducing contention for the link. A similar effect is evident in the RLP/OOS only case, where delay is accordingly increased due to UDP retransmissions. With the TCP oriented service also performing retransmissions however, UDP delay rises with increasing error rates. RLP has the smallest effect as it is the most economical but also the least efficient for TCP traffic, when considering both file transfer and WWW browsing. SR/M2 is slightly better than Snoop, except at the highest error rate where Snoop is not as effective for TCP traffic. These increases in delay reduce the up to 10% loss rate of the Default scheme to less than 2% when using the RLP/OOS scheme (see Figure 5.4). With two Cellular links in the Cellular/WAN2 scenario, the delay metrics shown in Figure 7.14 also show the Default scheme to be faster with higher loss rates,

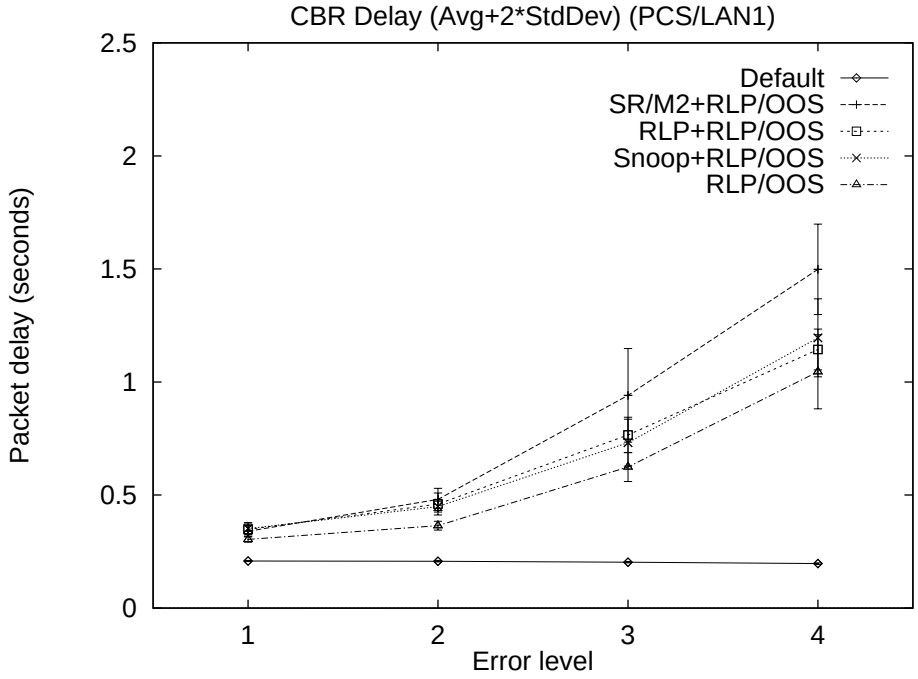

Figure 7.15: Real time conferencing delay over one multi service PCS link

again due to reduced contention from TCP traffic. When UDP frame retransmissions are added however in the RLP/OOS only case, delay increases with higher error rates, as there are two wireless links on the path, considerably increasing the amount of losses. Adding Snoop for TCP traffic has minimal effects as TCP performance is only slightly improved in this two wireless link topology (see Figures 7.2 and 7.8). RLP and SR/M2 on the other hand significantly improve the performance for both TCP applications, thus increasing contention on the link and the delay of the UDP application. SR/M2 is more effective for TCP, hence it has a larger effect on UDP delay. The increased delay reduces the up to 19% loss rate of the Default scheme to less than 4% with RLP/OOS (see Figure 5.6). Delay metric variance is very low with Cellular links.

With one PCS link in the PCS/LAN1 scenario, delay drops only slightly with higher error rates in the Default scheme, as shown in Figure 7.15. Introducing RLP/OOS for UDP traffic means increasing delay with higher error rates, mainly due to the long fades experienced on PCS links. This is consistent with the results obtained from single application tests. The additional delay introduced by the RLP and Snoop schemes and their retransmissions of TCP frames is relatively minor. SR/M2 has a more pronounced effect, as it is the best performing

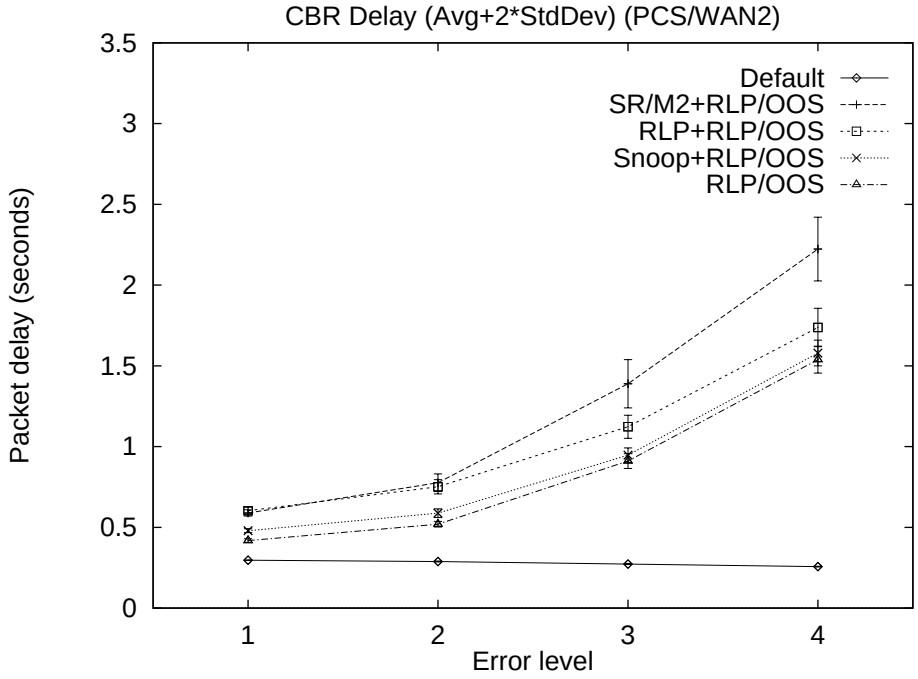

Figure 7.16: Real time conferencing delay over two multi service PCS links

scheme for both TCP applications (see Figures 7.3 and 7.9), hence increasing contention for the link. In all cases, when using RLP/OOS the loss rate for UDP is around 2%, compared to up to 10% for the Default scheme (see Figure 5.8). With two PCS links in the PCS/WAN2 scenario, the delay metrics are very similar but inflated for the higher path delay, as shown in Figure 7.16. Since Snoop is less effective with two wireless links, it is very close to RLP/OOS for UDP only, while RLP and SR/M2 introduce progressively more delay. As usual, this is a result of their improved TCP application performance (see Figures 7.4 and 7.10) which increases contention for the link. Loss is reduced from up to 18% with the Default scheme to 4% with RLP/OOS (see Figure 5.10). Delay metric variance is moderate due to the PCS link error model.

With one WLAN link on the path, as in the WLAN/LAN1 scenario, delay results are similar to the Cellular case, as shown in Figure 7.17. The Default scheme reduces its delay as TCP throughput drops, but the RLP/OOS scheme, although slower due to retransmissions, also reduces its delay for moderate to high loss rates. This is not unexpected as without any link layer enhancements in the WLAN/LAN1 scenario both TCP applications performed very bad (see Figures 7.5 and 7.11), leaving the link unused most of the time. In contrast, SR/M2,

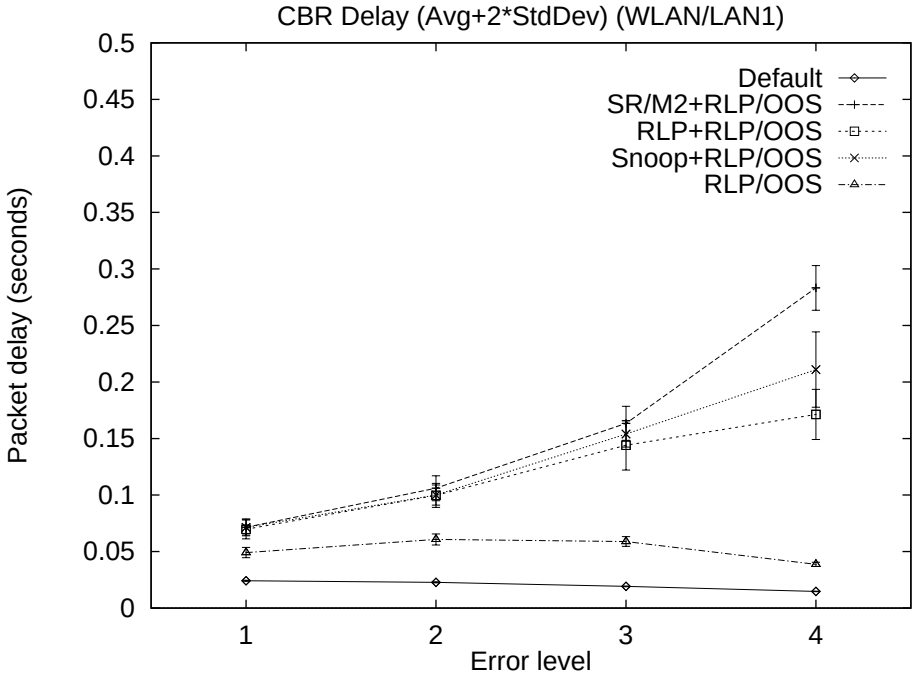

Figure 7.17: Real time conferencing delay over one multi service WLAN link

RLP and Snoop all considerably increased file transfer performance, with SR/M2 and RLP also significantly increasing WWW browsing performance. As a result, delay with them is much higher, with RLP having the smallest influence and SR/M2 the largest, mirroring their effect on TCP application performance. In this scenario the loss rate is reduced from up to 6% with the Default scheme to less than 0.5% with RLP/OOS (see Figure 5.12).

Finally, with two WLAN links in the WLAN/WAN2 scenario, the Default scheme again progressively exhibits smaller delays, as shown in Figure 7.18. RLP/OOS is relatively close due to the small propagation delay of WLAN links and the very bad performance of TCP applications in this scenario (see Figures 7.6 and 7.12). With Snoop for the TCP service the situation is nearly the same as with RLP/OOS only, as Snoop is not particularly effective in this topology. SR/M2 and RLP considerably increase performance for TCP applications, so they have a, nearly identical, effect on UDP delay. Compared to the single WLAN link scenario however, TCP applications achieve considerably lower performance due to the higher delay introduced by the WAN path. This explains why delay with SR/M2 as the TCP oriented service is higher with one link than with two links at high error rates. It also explains the lower delay metric variance

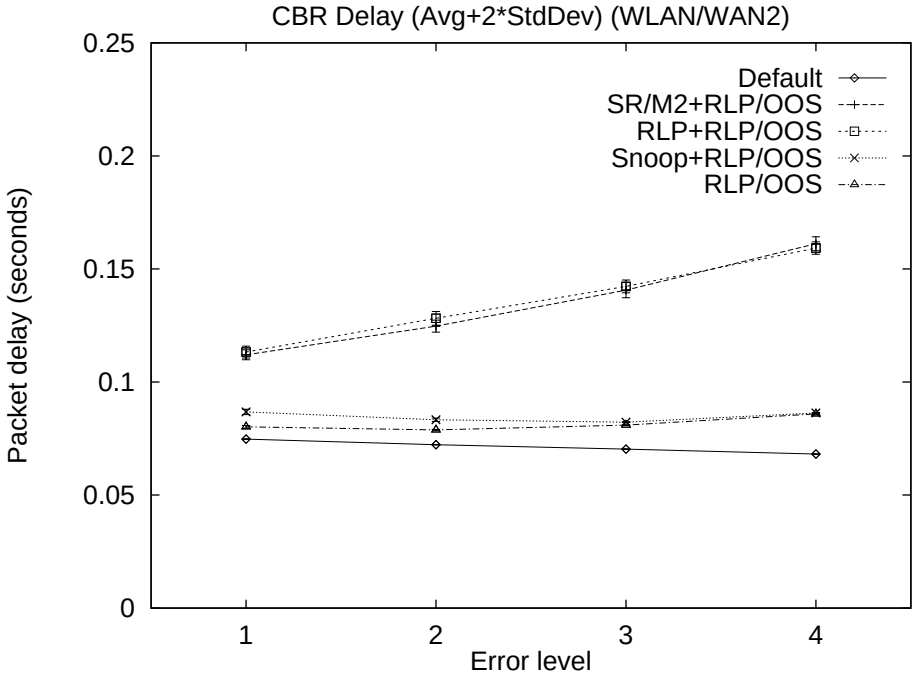

Figure 7.18: Real time conferencing delay over two multi service WLAN links

with this topology. In this scenario the loss rate is reduced from up to 11% with the Default scheme to less than 1% with RLP/OOS (see Figure 5.14).

7.4 Summary of Results

In this section we summarize our results by reviewing the performance of the various link layer schemes examined, discussing the interactions between them and drawing overall conclusions on the effectiveness of our multi service link layer proposal.

7.4.1 Link Layer Scheme Performance

In general, all link layer schemes tested performed similarly in the multi service (and application) tests as in their single service (and application) counterparts. SR/M2 was the most robust scheme for TCP traffic, especially at higher loss rates. RLP was more economical and efficient than SR/M2 at low error rates, but its performance degraded more rapidly with increasing error rates. The best choice between the two depended on the error rate and the type of link used

in each scenario. Snoop was effective only in unidirectional file transfers in one wireless link topologies only, failing to significantly improve WWW browsing performance in all tests. For TCP applications, SR/M2 improved file transfer throughput by 10-2500% and WWW browsing by 10-10000% over the Default scheme. Since the Default scheme was biased in favor of file transfer and against WWW browsing, file transfer with SR/M2 and RLP improved by a smaller factor in comparison to single application tests, while WWW browsing improved by a higher factor. Overall, TCP performance improved by virtually the same factor as in single application tests. In parallel, RLP/OOS reduced the loss rate of the real time conferencing application at the exact same low levels as with a single application on the path, allowing such applications to operate even under very harsh conditions.

The multi service link layer scheduler was very effective in providing the UDP application with the bandwidth it needed, also keeping its delay low. Contention between TCP and UDP traffic increased this delay, with a more serious impact from TCP oriented schemes that offered better TCP application performance. This occurred because UDP frames occasionally had to wait for TCP frames that were already being transmitted to finish. This is an unavoidable effect of sharing the link in practical systems: even when transmissions may be aborted (which is not always possible), doing so would waste a lot of wireless link bandwidth. While the least effective TCP oriented schemes had a reduced effect on UDP delay, the corresponding improvements in TCP performance more than balanced the scale in terms of overall application performance. RLP, which is more economical than both SR/M2 and Snoop, had a lesser impact on UDP delay even when it considerably improved TCP performance. It is possible that similar delay reductions could be achieved by using more economical full recovery schemes instead of SR/M2, for example the multiple acknowledgment schemes discussed in Section 4.2.

Both TCP applications used the same service, which meant that they had to share the available bandwidth by employing TCP congestion control mechanisms. This was complicated by wireless losses and the variable available bandwidth which depended on whether the UDP application was active or not. The Snoop scheme showed a bias against WWW browsing and in favor of unidirectional file transfer, since it was only able to improve TCP performance in a single direction, causing file transfer to become more aggressive and WWW browsing to back off. As a result, WWW browsing performance in some tests was *worse* than with the Default scheme. Although this could be avoided by using separate services for each application protected

by appropriate bandwidth allocations at the scheduler, it should be pointed out that the Snoop scheme was claimed to be an application independent solution that was supposed to enhance *any* TCP application [4], not just file transfer. On the other hand, the pure link layer SR/M2 and RLP schemes were not biased in favor of any application, being symmetric and location independent, hence they nearly always improved both file transfer and WWW browsing throughput, leading to a more fair sharing of the available bandwidth. This means that the use of separate services for each type of TCP application can be avoided, thus simplifying the multi service link layer and eliminating the need to determine appropriate rates for each such service.

7.4.2 Conclusions

Based on the multi service (and application) results and their analysis presented in this chapter, we have reached a number of conclusions regarding the effectiveness of our multi service link layer proposal.

- Link layer scheduling effectively limits the impact on UDP application delay and bandwidth caused by TCP retransmissions.
- By improving the efficiency of TCP oriented schemes we can further reduce their impact on the delay of other services.
- Either SR/M2 or RLP can be used to significantly improve TCP application performance in a multi service scenario.
- The best scheme to use for TCP applications depends on the error rate and error model of the underlying link.
- Location dependent schemes like Snoop can lead to unfairness and even degraded performance with simultaneous TCP applications.
- A single service using a pure link layer scheme like SR/M2 or RLP can enhance the performance of a mix of TCP applications.
- Overall, the multi service link layer scheme jointly improves UDP and TCP application performance by virtually the same factor as in single application tests.

Chapter 8

Interface with Higher Layers

This chapter shows how a multi service link layer can co-operate with existing and emerging higher layers. Section 8.1 describes a heuristic scheme for transparent mapping of application requirements to link layer services. Section 8.2 discusses the differentiated services model for *Quality of Service* (QoS) provision on the Internet and how it can be integrated with our approach. Section 8.3 describes an advanced interface to our multi service link layer scheme and how it may be used by emerging QoS aware higher layers.

8.1 Interface with the Existing Internet

A critical component of a multi service link layer is a function for mapping IP datagrams to services available on the link. The lowest common protocol layer on the Internet, the network layer, provides only a single service to higher layers, best effort IP datagram delivery, without any quality guarantees. A key assumption is that transport or even higher layer protocols can be used to improve this service to satisfy any additional application requirements. Our results however show that application performance over wireless links can be substantially enhanced by employing link layer schemes. Furthermore, due to the diversity of higher layer protocol and application requirements, a multi service link layer is needed to support existing and future application needs in an extensible manner. Since IP and the protocols using IP have no provisions for handling multiple services at the link layer though, we cannot rely on them to map their requirements to the services available on each link.

In order to maintain compatibility with the existing Internet infrastructure, our multi

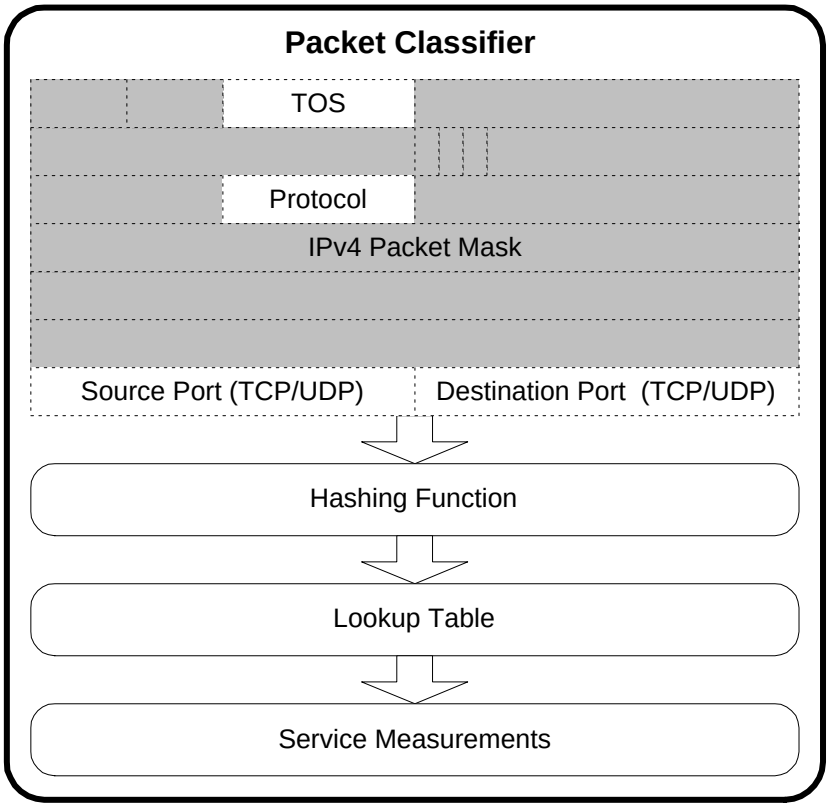

Figure 8.1: Heuristic packet classifier

service scheme should perform this mapping without requiring any changes to the interface of the link layer with other layers. Due to the scale of the Internet, such changes would take years to propagate everywhere, thus hindering deployment of our approach. In addition, performance should be *at least* as good as with a single service link layer, for *any* application: enhancing the performance of some applications should not degrade the performance of others. Applications should also never be mapped to services that degrade rather than improve their performance. This also eases deployment since it allows some services to be introduced immediately to selectively enhance performance for some traffic, without affecting the rest. New services can be added as new requirements and appropriate link layer schemes emerge.

Our link layer scheme has a single entry and exit point in each direction, to ease its integration with existing network protocol stacks (see Figure 6.1). Since no changes are allowed to this interface, the only information we can use for service selection is the incoming

IP datagrams themselves. In order to match application requirements with available services we can use a *heuristic classifier*. This classifier would employ heuristic rules to recognize certain applications based on IP, TCP and UDP header fields and assign their data to an appropriate service. All other data would use the *default* service, i.e. the raw service offered by the underlying link. Figure 8.1 shows data flow in this type of classifier for the current version of IP (IPv4). The headers of IP datagrams entering the classifier are masked to isolate the fields needed for classification (masked fields are grayed out). The result passes through a *hashing function* that produces an index to a *lookup table*, each entry of which points to one of the available services. Unrecognized applications are mapped to entries pointing at the default service. As a result, an application either uses a service enhancing its performance or the default service, which is no worse than with a single service link layer.

After packets are assigned to services, the classifier uses their size in order to implicitly deduce the share of the link allocated to each service. Over an implementation dependent time interval, the classifier should divide the amount of data assigned to each service by the total amount of data seen to get a fraction r_i, where $\sum_{i=1}^{n} r_i = 1$, for each of the n services. These fractions are used to update the *rate table* employed by the frame scheduler (see Figure 6.3). Regardless of whether higher layers perform packet scheduling or not, they expect a single service link layer to use the available bandwidth to send all data handed to it. For classification to be transparent, multiple services should share the link based on that (implicit) allocation, even though each service may introduce arbitrary amounts of error recovery overhead. This is achieved by measuring the fraction of data allocated to each service and employing these shares at the scheduler. Thus, unrecognized applications will be allocated at least the same amount of bandwidth that they would get in a single service scheme. Recognized applications on the other hand will trade off throughput, when the link is loaded, for better error recovery.

The header mask, hashing function and lookup table, or a single equivalent mapping function, are supplied by an external administrative module that is aware of application requirements, header fields and service properties. One field that can be used to select a service is the *Protocol* field, which contains standardized values to indicate TCP, UDP, or another protocol. All TCP applications can be mapped to a generic TCP enhancement service, implemented by one of the pure link layer schemes that we tested in Chapter 4. UDP applications on the other hand have varying requirements, so decisions should be made on a per application basis. For real time

playback applications we can use the services tested in Chapter 5. To recognize suitable UDP applications, we can look at the *source port* 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 a host without any previous arrangements. Thus, the protocol field along with well known ports can be used to recognize an application and map its data to the appropriate service. A final field that may be used is the *Type of Service* (TOS) field of the IP header, originally intended to specify application preferences and priorities [88]. 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 the relative priority of a packet. The TOS field is rarely used however, and it is currently being redefined to support differentiated services [86], as discussed in Section 8.2. A similar approach can be used to map IP version 6 (IPv6) headers to services [89], with the same reservations regarding the IPv6 *Traffic Class* field which is also being redefined to support differentiated services.

A significant drawback of heuristic classifiers is the amount of effort required to construct them. A manual matching process between applications and services must be performed for each link. Previous decisions must also be reviewed when new applications or services are added. The number of applications recognized is also an issue, 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. Although some QoS provisioning approaches use a combination of transport protocol, source/destination ports and source/destination host addresses to classify packets [90], they require end-to-end signaling to set up appropriate state at each intermediate node. Even if such signaling became common, maintaining separate state for each source/destination host pair would require large amounts of storage that are generally unavailable at the link layer.

A more important problem is that heuristic classifiers are complicated by IP security mechanisms that encrypt IP datagram payloads, and in particular the source and destination ports of TCP and UDP headers. Even worse, they replace the value of the protocol field with the identifier of the IP security protocol [41], leaving only the, rarely used, TOS field visible. As IP level security becomes more prevalent, classifiers will have to rely only on visible IP header fields, which are currently inadequate to describe application requirements.

8.2 Interface with Differentiated Services

In the following paragraphs we discuss the need to provide multiple services on the Internet, describe an emerging approach to this task, and show that it greatly complements our multi service link layer model.

8.2.1 Motivation for Internet Quality of Service

The single best effort service offered by IP is showing its limitations as more applications migrate from circuit switched networks with explicit delay guarantees, such as the telephone network, to the packet switched Internet. For example, the real time playback applications discussed in Section 5.1 have stringent delay requirements that cannot always be met due to congestion. For these applications to exploit the reduced costs offered by statistical multiplexing and still provide reasonable service to their users, it seems that some type of performance guarantees must be introduced on the Internet [62]. Users would presumably be willing to pay more for better Internet service if it would be cheaper to use the Internet rather than a circuit switched network for the same application. The main problem with QoS on the Internet is generally considered to be congestion, which translates to multiple problems from an application viewpoint. 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. Applications using transport protocols like UDP that do not provide error recovery have to deal with these losses themselves. For TCP based applications, such losses lead to even more increased delays due to end-to-end recovery.

In order to provide throughput guarantees, one approach is to use the *class based queueing* (CBQ) [75] scheme which reserves fractions of the available bandwidth for each *traffic class*. When there is more traffic than what the link can handle, CBQ ensures that each class does not use more than its allocated bandwidth share. What constitutes a traffic class and how classes should share the link is an administrative decision. Classification of IP datagrams into classes is based on the contents of their header fields, with each traffic class using a separate queue for its packets [76]. A packet scheduling mechanism is used to enforce link allocations, by selecting the traffic class that should transmit next every time the link becomes available. As long as

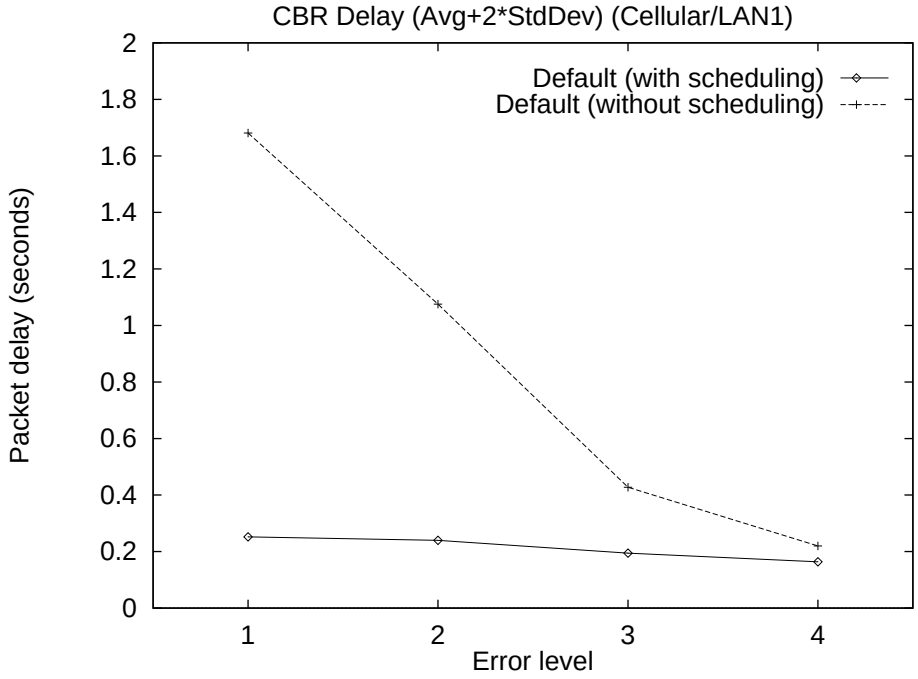

Figure 8.2: Delay with and without scheduling (one Cellular link)

a traffic class is not exceeding its allocation, packet scheduling, besides satisfying throughput requirements, also reduces packet delay and loss: packets do not have to wait in long common queues, and overflows leading to packet losses are avoided.

To assess the importance of scheduling, we repeated the multiple application tests of Chapter 7 in two different configurations. One test multiplexed all TCP and UDP applications over a single Default service, so that no scheduling or link enhancements were provided. This is how a single service link layer would operate without any link or IP level scheduling. The other test used two copies of the Default service, one for TCP and one for UDP, and allocated the link as in Chapter 7. Thus, enough bandwidth was reserved for UDP to satisfy the peak requirements of the real time conferencing application. In our tests persistent congestion was not an issue, since while the single UDP application transmits at a fixed rate when active, the two TCP applications back off when congestion appears. We disabled the nominal delay imposed to the IP layer by the link layer to allow packets to reach the frame scheduler and be queued there, so that in the two service case each protocol would use separate queues. In both cases real time conferencing achieved the same throughput, and of course loss rate, while the TCP applications

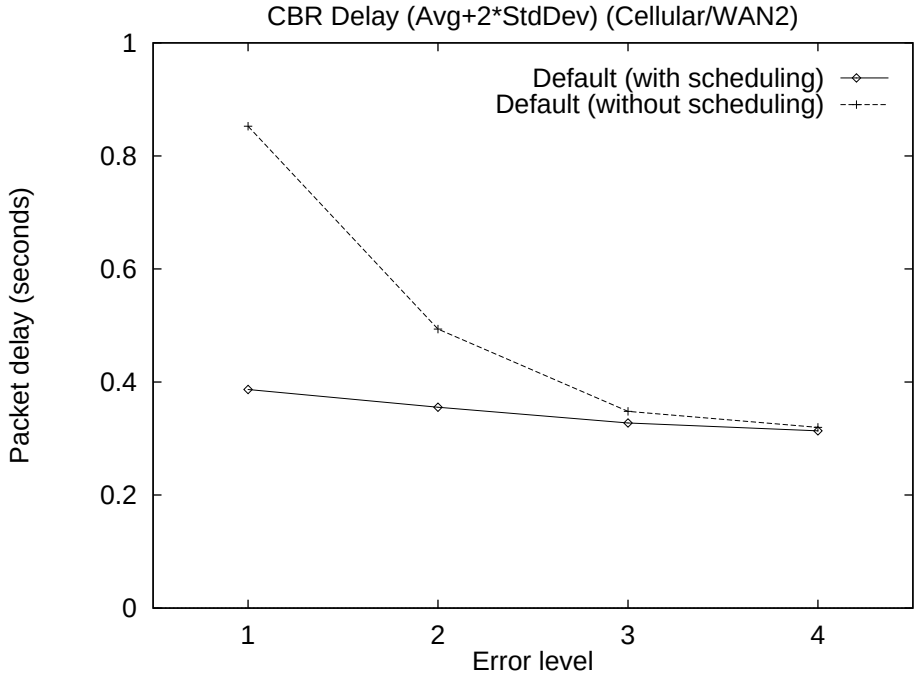

Figure 8.3: Delay with and without scheduling (two Cellular links)

themselves also had nearly the same performance. The main difference was on UDP application delay. Figure 8.2 shows average delay plus twice its standard deviation for the Cellular/LAN1 scenario and Figure 8.3 shows the same metric for the Cellular/WAN2 scenario. At higher error levels, delay without scheduling is similar since TCP performs very badly and therefore rarely contends with UDP. At low to moderate error levels however, contention leads to large increases in delay for the UDP application in the absence of scheduling.

The reason for this phenomenon is that during periods of UDP application inactivity, the two TCP connections increase their transmission rates so as to fully utilize the link. When the UDP application becomes active again, it finds a large amount of queued packets awaiting transmission. In the absence of scheduling, UDP packets have to wait behind TCP packets until the TCP connections back off due to the increased contention from the UDP application, thus increasing UDP packet delay. With two services on the other hand, UDP packets enter their own queue and are transmitted as often as needed to satisfy their service rates. The result is a nearly constant delay, slightly increased in the two wireless link scenario due to a longer end-to-end path. Very unexpectedly, without scheduling in the one wireless link scenario delays are

higher. This is because TCP has fewer losses to deal with and is therefore more aggressive, hence longer queues are formed and UDP packet delay increases even more. These conclusions are supported by results from tests with all other topologies and wireless links. These results imply that scheduling will eventually become integral to the Internet due to real time application requirements, hence our architecture was designed to be compatible with IP level scheduling.

8.2.2 Integrated and Differentiated Services

While scheduling mechanisms like CBQ are useful for isolated links, for example to provide controlled sharing of expensive links between the organizations paying for them [75], the end-to-end performance they provide is difficult to predict. End-to-end QoS characterization can be provided by restricting the senders to conform to a specific traffic generation profile, enforcing a common scheduling policy at all routers and performing admission control to ensure that the routers are not overloaded [77, 91]. Given these constraints, end-to-end delay bounds can be calculated [92, 93]. These bounds however are not very tight [94], making the provision of deterministic QoS guarantees expensive in terms of the amount of resources that must be reserved. An alternative is to provide multiple QoS levels to satisfy different types of requirements: a *guaranteed* service [95] providing strict delay bounds, a *controlled load* service [96] providing predictive delay bounds for adaptive applications, and a best effort service with no delay guarantees [62]. Guaranteed service is the most expensive and subject to strict admission control, but due to its relaxed delay bounds it does not fully utilize the link. Controlled load service costs less and has more relaxed admission requirements, but can only use bandwidth left over by the guaranteed service. Best effort service is the cheapest and has no admission control. A *resource reservation protocol* (RSVP) [97] must be used to handle admission control and resource reservations for each application [98].

This *integrated services* architecture for the Internet has been criticized in two ways. First, it must be deployed on large parts of the Internet to be useful, since end-to-end guarantees rely on performance at each router on a path. Second, reservations are made on a per *flow* basis, where a flow is defined as a stream of data between two user processes with the same QoS requirements. The problem with flows is that a huge number of them exists, severely limiting the scalability of any QoS scheme based on per flow state. In IPv6 for example a 20 bit *flow label* is defined as part of the IP header to identify flows between two hosts, while the host address

field is also expanded to 128 bits [89]. The only solution to this problem is aggregation of flow state, but it is not clear how this could be achieved in a large scale.

An alternative approach to QoS provision on the Internet that aims to avoid the limitations of the integrated services architecture is the *differentiated services* architecture [87]. The differentiated services model aggregates individual flows into a few classes either on their entrance to the network, or when they cross administrative domains. At these points only, flows may be rate limited, shaped or marked to conform to specific traffic profiles. These profiles are either negotiated between users and network providers (for aggregation on the entrance to the network) or between neighboring domains (for aggregation between domains). In both cases, traffic profiles and aggregation rules are only needed close to a host or at least within its domain. Inside a domain, each router only needs to select a *per hop behavior* (PHB) for each packet based on its class. The class is denoted by the 8 bit *differentiated services* (DS) field of the IP header, which subsumes both the IPv4 *type of service* field and the IPv6 *traffic class* field. State aggregation into a few classes means that this approach scales well, but the guarantees that may be provided are not as fine grained as with integrated services.

The architecture intentionally leaves the definition of PHBs and their implementations open, to allow experimentation with different schemes. For example, the *expedited forwarding* PHB is defined to provide a specific 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 [99]. This PHB provides very low delay and loss by eliminating congestion for a class, and may be implemented by many packet scheduling mechanisms, offering an end-to-end *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. In addition, 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 [100]. In this scheme, flows are *marked* with higher drop preference levels when they exceed the traffic profile for their class, rather than being rate limited as with expedited forwarding. Within each class, this scheme can be used to distinguish between more and less important traffic by marking packets with appropriate drop preferences at their entrance to the network [101]. This PHB may also be implemented by many scheduling mechanism and queue management schemes.

It is possible for different domains to support their own sets of PHBs, in which case the DS fields of packets crossing domain boundaries would need to be modified. Depending on the PHBs available, different end-to-end services may be offered. The services provided by this architecture are meant to offer various generic QoS levels as opposed to application specific guarantees, hence the decision to map traffic classes instead of flows to PHBs. Only entry points to a network must be aware of both application requirements and PHB semantics to perform flow aggregation into classes. Similarly, only entry points to domains must be aware of the semantics of PHBs available in their neighboring domains. Traffic policing, meaning rate limiting, shaping and marking, is also only performed at these points based on traffic profiles. For neighboring domains these profiles should be relatively static as they would represent large traffic aggregates, while at the entry points to the network they could be modified more frequently, depending on user requirements. This is far more economical than the integrated services approach that requires end-to-end signaling for each individual flow.

8.2.3 Differentiated Services and Multi Service Links

The differentiated services architecture and our multi service link layers are solutions to *orthogonal* problems. Differentiated services are concerned with congestion and its impact on throughput, delay and loss. The services provided are based on defining link independent PHBs that may be supported by appropriate packet scheduling and queue management mechanisms at the IP level. Error recovery mechanisms for wireless losses however are link dependent and their behavior cannot be standardized across heterogeneous links into a common set of PHBs. Multi service link layers on the other hand are concerned with wireless losses and how they should be managed for each type of application. The services provided are wireless link dependent and local. Although frame scheduling is provided, its goal is to protect services from each other by mirroring higher layer allocations rather than provide end-to-end guarantees.

At the same time, differentiated services and multi service link layers are *complementary* solutions. Providing differentiated services over wireless links may offer to applications a nominal IP level QoS, but their actual performance critically depends on wireless errors. It is not sufficient to reserve wireless link bandwidth for a TCP traffic class to guarantee acceptable performance, since only a small fraction of it can be used due to losses and inefficient TCP error recovery. This is clearly illustrated by our single TCP application measurements in Chap-

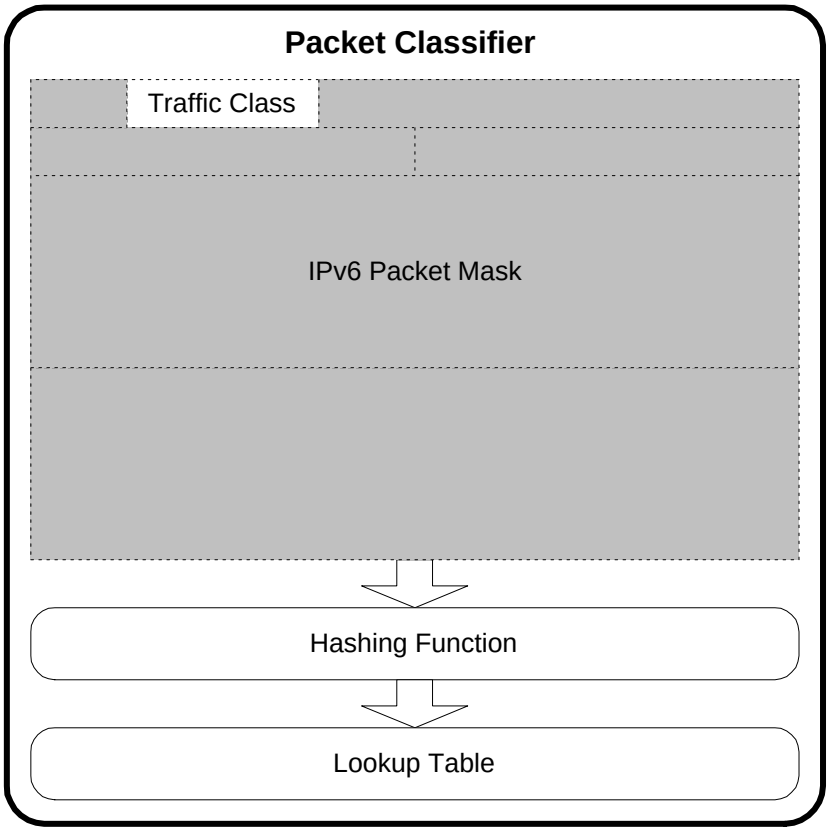

Figure 8.4: Differentiated services packet classifier

ter 4, where performance is not limited by congestion but by wireless errors. Similarly, our UDP application measurements in Chapter 5 show that applications may be unusable without error recovery. On the other hand, multi service link layers may provide adequate recovery to fully utilize wireless links, but they need higher layer guidance to perform scheduling. This is illustrated by Figures 8.2 and 8.3, where large delays for the real time conferencing application can only be avoided by allocating bandwidth according to end-to-end requirements that are not visible at the link layer. Hence, both solutions are needed to support enhanced end-to-end services.

We believe that differentiated services and multi service link layers are excellent complements to each other. Differentiated services are suitable for end-to-end congestion control, performing packet scheduling and queue management at each link, while multi service link layers can provide application dependent wireless error control, respecting higher layer scheduling

decisions despite the introduction of recovery overhead. They both offer a small number of services at each node (PHBs or link layer schemes) to support aggregated traffic classes with common requirements, rather than individual flows. They can be easily combined by extending the DS field to also specify the error requirements of each (sub)class of traffic. For example, a differentiated services traffic class with a reserved amount of bandwidth at each hop could be subdivided into two (four) subclasses with different error recovery requirements by using one (two) bits of the DS field. Each subclass would be in turn mapped to an appropriate link layer service. The administrative module choosing these mappings would only need to know how to match a few bits of the DS field, denoting generic application requirements, to the services available on the wireless link. These bits, along with the rest of the DS field, are only set at the points where flows are aggregated into classes. For example, applications could indicate their requirements when injecting their traffic to the Internet, with boundary routers translating them to local equivalents when administrative domains are crossed.

The result is a simplified multi service classifier, shown in Figure 8.4. Packet headers are masked to isolate the DS field, which is passed through the *hashing function* to produce an index into the *lookup table*, pointing at the appropriate service. The header mask, hashing function and lookup table, or a single equivalent mapping function, are supplied by the administrative module. Although Figure 8.4 shows the IPv6 header, where the DS field is the original *traffic class* field, the same procedure could be used for IPv4 headers, where the DS field is the original *type of service* field (see Figure 8.1). Unlike heuristic classifiers, the differentiated services classifier does not have to rely on multiple header fields and complex rules to determine application requirements, nor frequently update these rules to handle new applications. A single field is used instead, with well defined semantics. New applications may be mapped to existing classes if their requirements are not substantially different than those of previous applications, without changing the classifier. More importantly, the DS field is *not* hidden by IP security mechanisms, it is visible even in encrypted datagrams [41], unlike other header fields.

Subclasses of separate traffic classes that have the same error recovery requirements should be mapped to the same service, rather than separate instances of it. Since the differentiated services module performs scheduling and queue management, traffic entering the multi service link layer already obeys the corresponding sharing requirements. For example, two TCP traffic subclasses belonging to separate classes may be rate limited in different ways at the IP

level based on their bandwidth allocations. When they arrive 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. This means that regardless of the number of traffic classes, the multi service link layer only needs to maintain a single instance of each service. Setting the service rates can thus be achieved in two ways. If the subclasses of each traffic class have separate bandwidth allocations at the IP level, then the allocations for all subclasses mapped to the same service can be added to set its service rate at the link layer. In this case there is no need for a *service measurements* module, as shown in Figure 8.4. If subclasses however share a common bandwidth pool within each traffic class, then a service measurements module can be inserted after the lookup table as in Figure 8.1 to automatically determine service rates.

8.3 Advanced Quality of Service Interface

Our measurements in Chapters 4 and 5 show that depending on the error characteristics of each link, different link layer mechanisms are preferable. The QoS offered by these mechanisms depends not only on underlying link characteristics, but also on unpredictable environmental conditions. Due to these limitations, it is not possible to globally standardize a set of wireless link behaviors and expect them to be offered everywhere, as is the case with differentiated services PHBs. As a result, the performance characteristics of the services available at each wireless link will generally vary widely. In order for applications to select proper end-to-end services however, some kind of service characterization is required along network paths [102]. Besides verifying that an offered service is suitable for an application, end-to-end characterizations are also needed when path characteristics change significantly, forcing higher layer protocols or applications to modify their policies. Handoffs in cellular systems are an example of such drastic changes: *horizontal handoffs* may cause congestion and error characteristics to change, while *vertical handoffs* can dramatically alter wireless link characteristics and, consequently, end-to-end performance (see Section 2.1).

Since standardization of wireless link services and their behavior is not possible, in order to characterize the end-to-end service on a network path we must be able to dynamically *discover* what is offered at each wireless link. This can be achieved by having each service dynamically *characterize* its performance using a set of standardized *metrics*. Dynamic charac-

terization means that end-to-end performance may be evaluated as often as needed to take into account the unpredictability of wireless links. 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 or of underlying link characteristics. The same metrics, in the same units, should be offered by every service over every type of link. As a result, a common end-to-end QoS characterization module would be able to evaluate the local services offered at any wireless link and compose their local metrics into end-to-end ones.

We decided to use three link independent metrics reflecting the possible trade offs that can be made by each error recovery scheme: goodput, loss and delay. All these metrics are calculated dynamically over a measurement interval that could be the same as that used by the packet classifier to measure bandwidth allocations. The reported metrics could be smoothed by using a weighted average of the latest calculated value and the previous reported value, similar to the procedure employed by TCP to smooth delay estimates [27]. *Goodput* (g_i for service i) is defined as the ratio of higher layer data transmitted over the measurement interval to link layer data transmitted, including all types of overhead. The amount of higher layer data transmitted may be different than the amount received by the peer higher layer due to residual losses. The goodput metric can be calculated at the sender without any receiver feedback. Note the difference between this *link layer goodput* and *application goodput* as used in Chapter 4: the former includes both original application data and their (transport layer) retransmissions, while the latter only includes original application data. The reason is that the link layer cannot distinguish between original and retransmitted higher layer data.

To complement goodput, we report *loss* (l_i for service i), defined as the ratio of higher layer data lost to higher layer data transmitted (lost plus received). This is calculated by the receiver based on the sequence of data released to higher layers. It depicts the *residual* loss rate after error recovery. For full recovery ARQ schemes it will be zero, but for limited recovery ARQ and FEC schemes it may be larger than zero. The final metric is *delay* (d_i for service i), defined as the one way average delay (in seconds) for higher layer packets. Instead of using timestamps, it is easier, although less accurate, to estimate this metric at the receiver based on knowledge of the implemented scheme and wireless link characteristics. For example, ARQ schemes could estimate one way and round trip delay and add one round trip delay to their one way delay estimate for each retransmission. A FEC scheme could instead add the interval

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	$e_i = g_i * (1 - l_i)$	$\frac{\text{higher layer data received}}{\text{link layer data transmitted}}$

Table 8.1: Service characterization metrics

between loss of a recovered frame and its reconstruction from parity data to its one way delay estimate. Delay can thus be estimated by the receiver without adding any per frame overhead.

These link independent metrics can be used by higher layer modules in various ways. The delay metric denotes the error recovery delay of a service, since there is no congestion inside the multi service link layer. It can be added to the queueing delay of the IP level scheduler to provide a total delay for each node, which can in turn 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 also falls within their tolerance limits. Since goodput is calculated by the sender only, loss has to be combined with it to provide an *effective goodput* metric, defined as $e_i = g_i * (1 - l_i)$ for service i. This denotes the ratio of higher layer data received to link layer data transmitted. Essentially, e_i shows how much of the bandwidth allocated to service i is used for data actually received by the peer, after discounting error recovery overhead and residual losses. All these metrics are summarized in Table 8.1.

Effective goodput can be used to estimate the throughput offered by a service. Assuming exclusive use of the link bandwidth B, the throughput of service i would be $B * e_i$. If service i is allocated a service rate r_i in the frame scheduler (which mirrors the allocations of the IP level scheduler), its throughput would be instead $B * r_i * e_i$. This formula may be used to estimate the throughput for each service given *any* allocation of service rates, or to calculate the service rate needed to achieve a target throughput for a particular service. This flexibility shows the advantages of using goodput instead of throughput for service characterization: goodput is independent of the bandwidth allocated to a service in the past, hence it can be used to predict

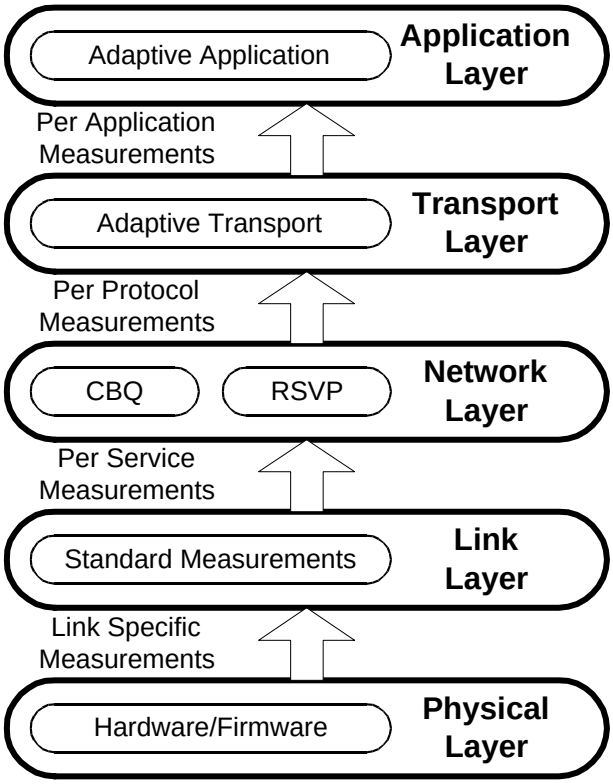

Figure 8.5: Propagation of service measurement feedback

its behavior with different allocations.

The metrics reported by each service may be used at multiple layers to serve different needs, as shown in Figure 8.5. At the lowest level, the physical layer provides hardware dependent information (such as fixed one way delay) that may be used by the link layer services along with their own mechanisms 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 appropriate bandwidth allocations for each service. End-to-end QoS schemes may use signaling protocols like RSVP to gather information about the services available at each node in order to estimate end-to-end service characteristics. These path characteristics can be used by transport protocols to adapt their operation to prevailing conditions. For example TCP could limit its congestion window to respect available bandwidth limitations, instead of alternating between overloading the link and backing off after congestion appears. Applications may also use end-to-end characteristics to adapt their operation. For example video conferencing applications may select encoding

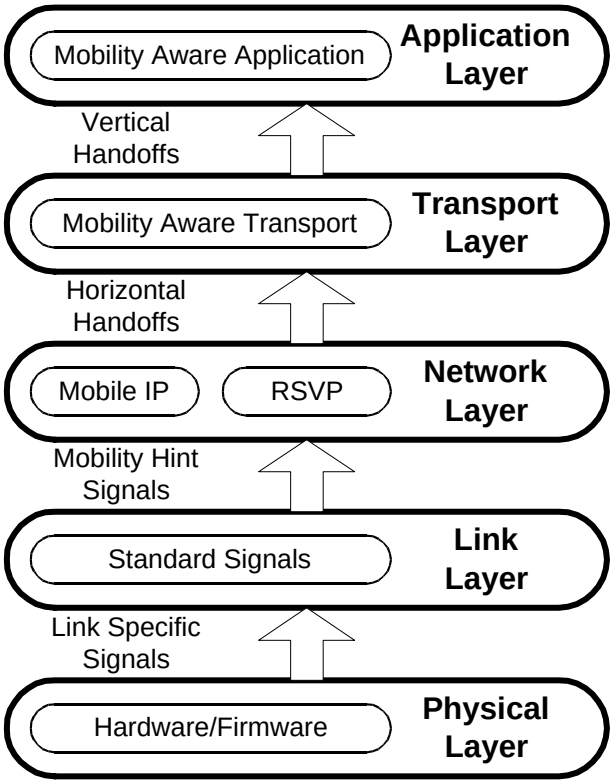

Figure 8.6: Propagation of mobility feedback

schemes that are sufficiently robust to deal with the residual loss rates of the underlying path. Service metrics can be *refined* at each layer to better serve higher layer needs, as in the network layer where local metrics are used to compose end-to-end metrics.

To help higher layers deal with handoffs and mobility in general, we can extend this interface to provide *mobility hints* to interested parties via *upcalls* [103], as shown in Figure 8.6. The link layer can use hardware registers or signals and combine them with its own state to detect significant events such as connections and disconnections. Each such event is a mobility hint: if a handoff is taking place, higher layers will receive one disconnection and one connection upcall for different links. If on the other hand the upcalls show that the same link is used, the disconnection was due to adverse error conditions rather than handoffs. These link independent upcalls can be used by the IP mobility extensions [33] to allow fast detection of handoffs, instead of relying on periodic network layer probes. IP mobility extensions can distinguish between actual handoffs and false alarms using the IP address of their peer across the link. Higher layers

may be notified by the network layer of horizontal and vertical handoffs via further upcalls. For example, TCP may be notified of pending horizontal handoffs to temporarily freeze its timers and avoid timeouts during disconnection intervals [38, 39]. In turn, applications may be notified of vertical handoffs that dramatically change path characteristics. For example, a video conferencing application could change the encoding scheme used to a higher or lower resolution one to adapt to the available bandwidth on the new path [104]. In this example, the application would discover these new characteristics by using end-to-end QoS provisioning mechanisms to query the metrics exported by services on the new wireless link.

8.3.1 Related Research

Our QoS interface was influenced by the needs of resource reservation mechanisms such as *One Pass With Advertising* (OPWA) [102]. This approach attempts to avoid the limitations of previous *one pass* and *two pass* resource reservation schemes. One pass schemes cannot specify a desired service in advance as they do not know what is available on a path [97]. As a result, the resources reserved may provide inadequate service. Two pass schemes specify a service in advance but make very restrictive reservations on the first pass to ensure that it will be provided, relaxing them on a second pass [105]. Such reservations may fail due to tight restrictions on the first pass. In the OPWA scheme, an *advertising* pass is first made to discover information on the services available on the path, and then a reservation pass actually reserves the resources needed for the selected end-to-end service. Our interface allows OPWA schemes to discover throughput, delay and loss restrictions imposed by wireless links on end-to-end paths, by looking at local multi service metrics. After that information is gathered, applications may decide which of the possible end-to-end services is best for their needs. The reservation pass can then set up appropriate mapping state or service rates to provide the requested service.

We decided to provide mobility hints to notify higher layers that they should revise their end-to-end path characterizations after handoffs, in order to satisfy the demands of mobility aware applications that can adapt their bandwidth demands based on resource availability [104]. The metrics provided by the new link after a handoff can be used to update path characteristics. A similar proposal has been made in the context of IP mobility extensions: a mobility aware router could notify remote hosts communicating with a wireless host of new link characteristics after a handoff [106]. This approach however does not support dynamic link characterization

between handoffs or multiple link layer services, thus being very limited compared to our approach. Another proposed adaptation interface for mobility aware applications allowed each application to request to be notified when the bandwidth available to it deviated from a specified range [107]. An application supporting multiple encodings of its data stream would select an appropriate bandwidth range for each, and switch from one to the other whenever the available bandwidth moved into another range. This interface is not appropriate for the link layer however, as it requires end-to-end signaling and maintenance of large amounts of per application state. By using our mobility detection hints however, this interface may easily be implemented at higher layers. A QoS management module would provide that interface at each host, accepting bandwidth range requests from local applications. Instead of constantly monitoring each path, it would only do so after receiving a mobility notification. After updating its path characterizations, it would check its data base and notify all applications whose bandwidth ranges had changed.

Chapter 9

Conclusions and Future Research

This chapter presents the conclusions drawn from our research and possible future extensions to our work. Section 9.1 summarizes the dissertation and identifies our original contributions. Section 9.2 discusses directions for further research on the interaction between multi service link layers and end-to-end Quality of Service provisioning schemes.

9.1 Summary and Contributions

Our work differs from other efforts to improve Internet application performance over wireless links largely due to our *far* more extensive measurements. Previous studies concentrated on enhancing the performance of file transfers over TCP, using network topologies containing a single wireless link of a specific type. We instead simulated three different applications with diverse requirements, using both TCP and UDP, over topologies with one or two wireless links, employing three wireless links with varying characteristics and error models. Our results showed that file transfer is an inadequate model for both UDP *and* TCP applications. In addition, the most popular application on the Internet, World Wide Web browsing, is only optimized when performance in *both* traffic directions is enhanced. This shows the inherent limitations of unidirectional error recovery schemes. Furthermore, the best solution for each type of link depends on its characteristics, hence end-to-end approaches are inferior to local ones for *any* path containing heterogeneous wireless links.

A critical conclusion from our single application tests was that applications with diverse requirements are best served by *fundamentally* different error recovery schemes, hence a

link layer offering a single service *cannot* be considered a universal solution. To improve the performance of arbitrary mixes of applications, we proposed a *multi service link layer* architecture that supports the simultaneous operation of multiple error recovery schemes, each targeted to different higher layer needs. A multi service link layer consists of a set of *link independent* components that provide service isolation and controlled link sharing for *any* type of service, and an arbitrary number of *link dependent* mechanisms implementing those services. The link independent components can be reused for any type of link while the link dependent mechanisms are simple modifications of existing error recovery schemes customized for the underlying wireless link.

To evaluate this scheme, we repeated our single application tests with all applications simultaneously operating over multi service links, each using the service best suited to its requirements. Our results showed that despite the presence of multiple independent services, the performance gains for each application were *virtually identical* to those achieved when operating in isolation over the same service, scaled for bandwidth limitations due to sharing. Hence, the multi service link layer approach offers a *universal* solution that can be easily extended with additional services to support new application requirements and ported to any wireless link with minimal effort. To transparently integrate our approach with the Internet, we showed how it can operate by heuristically mapping traffic classes to appropriate services without *any* changes to other protocols. Then we showed how it can co-operate with the *Differentiated Services* architecture for Quality of Service provisioning so as to combine congestion and wireless error management based on end-to-end requirements. Finally, we described a link and service independent interface to our scheme that allows service properties to be discovered dynamically and end-to-end services to be composed over heterogeneous network paths.

To further clarify our contributions, let us review the questions posed in Chapter 1 regarding the scope of our work, along with the detailed answers provided by our research.

- *Where in the protocol stack should we concentrate?* Previous research has attacked Internet performance problems over wireless links at either the transport or the link layer level. We argued in Chapter 2 that link layer solutions are applicable to any link and network topology and are easier to deploy due to their local nature. Our TCP results in Chapter 4 showed that performance can be tremendously improved by link layer schemes without *any* changes to higher layers. Our UDP results in Chapter 5 also showed that

similar gains can be achieved for applications using unreliable transport protocols. In the absence of link layer improvements, such applications would not even be able to operate over wireless links without individually modifying each application for *all* types of links.

- *Does a single link layer work for all applications?* Previous research has concentrated on improving unidirectional file transfer performance over TCP. By testing both file transfer and WWW browsing in Chapter 4 we showed that file transfer is *not* an adequate model for interactive applications such as WWW browsing, the most popular application on the Internet. Our results showed that the unidirectional Snoop scheme that was considered appropriate for any TCP application is actually *unable* to optimize WWW browsing performance. We also simulated a real time conferencing application using UDP in Chapter 5. Our results showed that its performance is optimized with a novel mechanism that we proposed which is *fundamentally* different than those appropriate for TCP. Our results thus show that different applications are best served by different link layer mechanisms.

- *Do we need to violate layering to optimize performance?* Some researchers have argued that link layer schemes for wireless links *must* be aware of higher layer, and in particular TCP, semantics to optimize performance by avoiding conflicts between link and higher layer error recovery. We show that previous measurements supporting this argument are *not* representative of realistic link layer schemes. Our results in Chapters 4 and 5 show instead that link layer schemes can optimize performance without *any* awareness of higher layer *or* application semantics. We also showed that the transport layer aware Snoop scheme is *unable* to enhance performance over paths including multiple wireless links due to its reliance on end-to-end acknowledgments. Our results thus show that layering violations are unnecessary and may even severely limit the applicability of a scheme.

- *Does the same link layer work for all links?* In contrast to previous studies that were limited to a single type of link, our tests considered a wide range of wireless links and error models. Our TCP application measurements in Chapter 4 showed that the best scheme for each wireless link depends on the underlying error model and error level: limited recovery is more efficient for light error conditions while full recovery is preferable for harsher conditions. Our UDP application measurements in Chapter 5 similarly showed that choosing between FEC and ARQ schemes depends on underlying link parameters:

FEC may be preferable for high delay links while ARQ is more appropriate for low delay links and harsh error conditions. Our results thus show that link independent end-to-end modifications *cannot* match the performance improvements provided by local link layer schemes, especially over heterogeneous network paths.

- *How can multiple schemes be combined over a single link?* Since our results showed that different link layer mechanisms are suitable for applications with diverse requirements, a single link layer scheme cannot be considered a *universal* solution. In Chapter 6 we proposed a *multi service link layer* architecture that combines simple mechanisms to support multiple higher layer requirements in parallel. Our scheme can accommodate *any* number of *arbitrary* link layer mechanisms, each targeted to a different class of traffic. Services are isolated and protected from each other by a reusable set of link independent modules. Each service can employ *any* mechanism appropriate to its goals and be heavily customized for the underlying link. Existing link layer schemes can be used to implement these services with minimal modifications.

- *How well does a multi service link layer perform?* In Chapter 7 we evaluated our proposal by repeating the tests of Chapters 4 and 5 with all applications operating simultaneously over multi service link layers. Each application employed the service best suited to its needs. Our results showed that despite the presence of multiple services over each wireless link, *all* applications improved their performance by virtually the same factor as when operating in isolation over the same link layer mechanism, scaled for the bandwidth limitations imposed by link sharing. Our results thus show that diverse applications can *simultaneously* optimize their performance over wireless links using our multi service link layers. Since our architecture can be easily extended with additional services and ported to any wireless link, it is a *universal* solution.

- *How does a multi service link layer interact with higher layers?* To ease deployment of our architecture on the Internet we developed a heuristic approach for mapping higher layer data to link layer services based on application requirements. This scheme optimizes the performance of recognized applications without *any* changes to existing protocols. In order to support end-to-end Quality of Service (QoS) requirements, we showed how our architecture can complement the *Differentiated Services* model for QoS provisioning on the

Internet to provide combined congestion and wireless error management to applications. We also developed a link and service independent service characterization and mobility detection interface that can be used to dynamically compose end-to-end services based on the local services available at each wireless link of a network path.

9.2 Future Research

One direction for future research on multi service link layers involves further extensions to our testing environment. With respect to TCP applications, although file transfer and WWW browsing seem to be quite general as models of unidirectional and bidirectional applications, respectively, further models of interactive applications like Telnet [52] or non interactive but bidirectional ones like POP [53] could shed further light on the services best suited to them. UDP applications on the other hand are considerably more varied in their requirements, and our real time conferencing model can only be considered adequate for a particular class of playback applications [62]. By testing more UDP based application models we would be able to add more services to our multi service link layer scheme, thus enlarging the range of applications that can benefit from our enhancements. Additional error recovery schemes, using both ARQ and FEC techniques, could be tested in an attempt to improve service efficiency and performance. Finally, more wireless error models should be used with all those applications and recovery schemes to allow us to port our scheme to other types of wireless links.

Another research direction is to further examine the interactions between our local multi service scheme and end-to-end Quality of Service (QoS) provisioning schemes. Various issues arise with respect to the integration of our scheme with the *Differentiated Services* model for the Internet [87]. The DS field of the IP header [86] must be extended to indicate error recovery requirements so as to ease the mapping of higher layer traffic to link layer services. The interactions between link layer services and frame scheduling and the various *per hop behaviors* defined for the differentiated services model should also be investigated [99, 100]. This involves first determining local performance at each node and then extending these characterizations to the services offered over end-to-end paths. This would be an ongoing task since the differentiated services model allows new per hop behaviors to be defined as needed and leaves open the packet scheduling and queue management mechanisms used to implement them [87]. A link

independent programming interface should also be defined to allow the IP level differentiated services module to instruct the link layer how it should map DS field values to available services and schedule link resources among services.

A related topic involves the composition of end-to-end services using our link and service independent Quality of Service interface for dynamic service characterization. Resource reservation schemes such as *one pass with advertising* [102] can be used to discover the services available at each link and combine them into a set of available end-to-end application oriented services. One issue arising in this setting is how these local descriptions can be aggregated to provide end-to-end characterizations that are both compact and expressive. Another issue is avoiding a combinatorial explosion on the number of end-to-end services available over paths with multiple wireless links. A service classification scheme could be developed to allow the reservation module to avoid combining services that are inherently incompatible. Such a scheme should rely on generic characteristics of each service, for example full or limited reliability, rather than on current performance metrics over any given wireless link.

The performance of end-to-end services based on our multi service link layers is inherently dynamic due to the unpredictable nature of wireless links. Handoffs in cellular systems can cause even long term performance to change. Future higher layers and applications could use dynamic performance metrics to adapt their operation based on prevailing conditions. In order to complement our dynamic service characterization metrics we developed a link and service independent mobility interface to help higher layers detect mobility and update end-to-end metrics. One issue with this scheme is the need for propagation of local mobility indications towards distant hosts communicating over the link. Another issue is the division of mobility handling tasks between multiple adaptive layers. A more general issue is how each protocol and application could react to mobility and dynamic end-to-end conditions. We have listed some preliminary ideas in this direction when presenting our service characterization and mobility detection interfaces.

Bibliography

[1] "GSM world newsletter," Available from `http://www.gsm.org`, 1999.

[2] "CDMA worldwide," Available from `http://www.cdg.org`, 1999.

[3] J. E. Padgett, C. G. Günther, and T. Hattori, "Overview of wireless personal communications," *IEEE Communications Magazine*, vol. 33, no. 1, pp. 28–41, January 1995.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, December 1997.

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

[6] B. Tuch, "Development of WaveLAN, an ISM band wireless LAN," *AT&T Technical Journal*, vol. 72, no. 4, pp. 27–37, July/August 1993.

[7] D. Duchamp and N. F. Reynolds, "Measured performance of a wireless LAN," in *Proceedings of the 17th IEEE Conference on Local Computer Networks*, September 1992, pp. 494–499.

[8] "WaveLAN IEEE specifications," Available from `http://www.wavelan.com`, 1999.

[9] D. Eckhardt and P. Steenkiste, "Measurement and analysis of the error characteristics of an in-building wireless network," in *Proceedings of the ACM SIGCOMM '96*, August 1996, pp. 243–254.

[10] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proceedings of the Winter Simulation Conference*, December 1996, pp. 597–604.

[11] G. Xylomenos and G. C. Polyzos, "TCP and UDP performance over a wireless LAN," in *Proceedings of the IEEE INFOCOM '99*, March 1999, pp. 439–446.

[12] B. P. Crow, I. Widjaja, J. Geun Kim, and P. T. Sakai, "IEEE 802.11 wireless local area networks," *IEEE Communications Magazine*, vol. 35, no. 9, pp. 116–126, September 1997.

[13] G. Bao, "Performance evaluation of TCP/RLP protocol stack over CDMA wireless link," *Wireless Networks*, vol. 2, no. 3, pp. 229–237, 1996.

[14] T. Alanko, M. Kojo, H. Laamanen, M. Liljeberg, M. Moilanen, and K. Raatikainen, "Measured performance of data transmission over cellular telephone networks," *Computer Communications Review*, vol. 24, no. 5, pp. 24–44, October 1994.

[15] S. Lin, D. J. Costello, and M. J. Miller, "Automatic-repeat-request error-control schemes," *IEEE Communications Magazine*, vol. 22, no. 12, pp. 5–17, December 1984.

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

[17] W. Simpson, "The point-to-point protocol (PPP)," Internet Request For Comments, July 1994, RFC 1661.

[18] A. DeSimone and S. Nanda, "Wireless data: Systems, standards, services," *Wireless Networks*, vol. 1, no. 3, pp. 241–253, 1995.

[19] J. Cai and D. J. Goodman, "General packet radio service in GSM," *IEEE Communications Magazine*, vol. 35, no. 10, pp. 122–131, October 1997.

[20] E. Dahlman, B. Gudmundson, M. Nilsson, and J. Sköld, "UMTS/IMT-2000 based on Wideband CDMA," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 70–80, September 1998.

[21] D. J. Goodman, "Trends in cellular and cordless communications," *IEEE Communications Magazine*, vol. 29, no. 6, pp. 31–40, June 1991.

[22] "Iridium system technology," Available from `http://www.iridium.com`, 1999.

[23] R. H. Katz and E. A. Brewer, "The case for wireless overlay networks," *Proceedings of the SPIE–The International Society for Optical Engineering*, vol. 2667, pp. 77–88, 1996.

[24] J. Postel, "Internet protocol," Internet Request For Comments, September 1981, RFC 791.

[25] J. Postel, "User datagram protocol," Internet Request For Comments, August 1980, RFC 768.

[26] J. Postel, "Transmission control protocol," Internet Request For Comments, September 1981, RFC 793.

[27] V. Jacobson, "Congestion avoidance and control," in *Proceedings of the ACM SIGCOMM '88*, August 1988, pp. 314–329.

[28] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," Internet Request For Comments, January 1997, RFC 2001.

[29] K. Fall and S. Floyd, "Simulation based comparisons of Tahoe, Reno and SACK TCP," *Computer Communications Review*, vol. 26, no. 3, pp. 5–21, July 1996.

[30] B. Gallaghan, B. Pawlowski, and P. Staubach, "NFS version 3 protocol specification," Internet Request For Comments, June 1995, RFC 1813.

[31] V. Jacobson, "Compressing TCP/IP headers for low-speed serial links," Internet Request For Comments, February 1990, RFC 1144.

[32] R. Cáceres and L. Iftode, "Improving the performance of reliable transport protocols in mobile computing environments," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 850–857, June 1995.

[33] C. Perkins, "IP mobility support," Internet Request For Comments, October 1996, RFC 2002.

[34] P. Manzoni, D. Ghosal, and G. Serazzi, "Impact of mobility on TCP/IP: An integrated performance study," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 5, pp. 858–867, June 1995.

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

[36] R. Yavatkar and N. Bhagawat, "Improving end-to-end performance of TCP over mobile internetworks," in *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, December 1994, pp. 146–152.

[37] K. Ratnam and I. Matta, "WTCP: an efficient mechanism for improving TCP performance over wireless links," in *Proceedings of the 3rd IEEE Symposium on Computers and Communications*, June 1998, pp. 74–78.

[38] K. Brown and S. Singh, "M-TCP: TCP for mobile cellular networks," *Computer Communications Review*, vol. 27, no. 5, pp. 19–43, October 1997.

[39] B. S. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan, "Improving performance of TCP over wireless networks," in *Proceedings of the 17th International Conference on Distributed Computing Systems*, May 1997, pp. 365–373.

[40] K. Brown and S. Singh, "M-UDP: UDP for mobile celullar networks," *Computer Communications Review*, vol. 26, no. 5, pp. 60–78, October 1996.

[41] S. Kent and R. Atkinson, "IP encapsulating security payload (ESP)," Internet Request For Comments, November 1998, RFC 2406.

[42] H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *Wireless Networks*, vol. 1, no. 4, pp. 469–481, 1995.

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

[44] E. A. Brewer, R. H. Katz, Y. Chawathe, S. D. Gribble, T. Hodes, G. Nguyen, M. Stemm, T. Henderson, E. Amir, H. Balakrishnan, A. Fox, V. N. Padmanabhan, and S. Seshan, "A network architecture for heterogeneous mobile computing," *IEEE Personal Communications*, vol. 5, no. 5, pp. 8–24, October 1998.

[45] "UCB/LBNL/VINT Network Simulator - ns (version 2)," Documentation and source code available from `http://www-mash.cs.berkeley.edu/ns`, 1999.

[46] T. R. Henderson and R. H. Katz, "Transport protocols for Internet-compatible satellite networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 2, pp. 326–344, February 1999.

[47] G. Xylomenos, "Multi Service Link Layer (MSLL) support for ns-2," Available from `http://www-cse.ucsd.edu/users/xgeorge`, 1999.

[48] C. A. Kent and J. C. Mogul, "Fragmentation considered harmful," in *Proceedings of the ACM SIGCOMM '87 Workshop on Frontiers in Computer Communications*, August 1987, pp. 314–329.

[49] J. Postel and J. Reynolds, "File transfer protocol (FTP)," Internet Request For Comments, October 1985, RFC 959.

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

[51] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol – HTTP/1.1," Internet Request For Comments, June 1999, RFC 2616.

[52] J. Postel and J. Reynolds, "TELNET protocol specification," Internet Request For Comments, May 1983, RFC 854.

[53] J. Myers and M. Rose, "Post office protocol - version 3," Internet Request For Comments, November 1994, RFC 1725.

[54] P. S. Yu and S. Lin, "An efficient selective-repeat ARQ scheme for satellite channels and its throughput analysis," *IEEE Transactions on Communications*, vol. 29, no. 3, pp. 353–363, March 1981.

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

[56] M. C. Chuah, B. Doshi, S. Dravida, R. Ejzak, and S. Nanda, "Link layer retransmission schemes for circuit-mode data over the CDMA physical channel," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 195–211, 1997.

[57] A. Al-Zoman, J. DeDourek, and B. Kurz, "Automatic retransmission rather than automatic repeat request," in *Proceedings of the 1994 International Conference on Network Protocols*, October 1994, pp. 48–55.

[58] E. Ayanoglu, S. Paul, T. F. LaPorta, K. N. Sabnani, and R. D. Gitlin, "AIRMAIL: a link-layer protocol for wireless networks," *Wireless Networks*, vol. 1, no. 1, pp. 47–60, 1995.

[59] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, October 1995.

[60] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgment options," Internet Request For Comments, October 1996, RFC 2018.

[61] S. Floyd, "TCP and explicit congestion notification," *Computer Communications Review*, vol. 24, no. 5, pp. 8–23, October 1994.

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

[63] M. Khansari, A. Jalali, E. Dubois, and P. Mermelstein, "Low bit-rate video transmission over fading channels for wireless microcellular systems," *IEEE Transactions on Circuits and Systems*, vol. 6, no. 1, pp. 1–11, February 1996.

[64] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, November 1994, pp. 604–612.

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

[66] H. Liu, H. Ma, M. El Zarki, and S. Gupta, "Error control schemes for networks: an overview," *Mobile Networks and Applications*, vol. 2, no. 2, pp. 167–182, 1997.

[67] S. Lin and P. S. Yu, "Hybrid ARQ scheme with parity retransmission for error control of satellite channels," *IEEE Transactions on Communications*, vol. 30, no. 7, pp. 1701–1719, July 1982.

[68] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Transactions on Communications*, vol. 36, no. 4, pp. 389–400, April 1988.

[69] A. J. McAuley, "Reliable broadband communication using a burst erasure correcting code," in *Proceedings of the ACM SIGCOMM '90*, September 1990, pp. 297–306.

[70] P. Godlewski, M. Braneci, and A. Serhrouchni, "An error control scheme for multicast communications over an ATM network," in *Proceedings of the Singapore ICCS '94*, November 1994, pp. 305–309.

[71] G. Xylomenos and G. C. Polyzos, "Enhancing wireless Internet links," in *Proceedings of the 1998 International Conference on Telecommunications*, June 1998, pp. 394–398.

[72] G. C. Polyzos and G. Xylomenos, "Enhancing wireless Internet links for multimedia services," in *Proceedings of the 5th International Workshop on Mobile Multimedia Communications*, October 1998, pp. 379–384.

[73] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," *Proceedings of the IEEE*, vol. 83, no. 10, pp. 1374–1396, October 1995.

[74] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[75] S. Floyd and V. Jacobson, "Link-sharing and resource management models for packet networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 4, pp. 365–386, August 1995.

[76] I. Wakeman, A. Ghosh, J. Crowcroft, V. Jacobson, and S. Floyd, "Implementing real time packet forwarding policies using streams," in *Proceedings of the 1995 Usenix Winter Technical Conference*, January 1995, pp. 71–82.

[77] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344–357, June 1993.

[78] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proceedings of the ACM SIGCOMM '89*, September 1989, pp. 3–12.

[79] J. C. R. Bennett and H. Zhang, "WF^2Q: worst-case fair weighted fair queueing," in *Proceedings of the IEEE INFOCOM '96*, March 1996, pp. 120–128.

[80] S. Golestani, "A self-clocked fair queueing scheme for broadband applications," in *Proceedings of the IEEE INFOCOM '94*, June 1994, pp. 636–646.

[81] T. Chen, P. Krzyzanowsi, M. R. Lyu, C. Sreenan, and J. Trotter, "A VC-based approach for renegotiable QoS in wireless ATM networks," in *Proceedings of the 6th IEEE International Conference on Universal Personal Communications*, October 1997, pp. 119–123.

[82] G. Brasche and B. Walke, "Concepts, services, and protocols of the new GSM Phase 2+ general packet radio service," *IEEE Communications Magazine*, vol. 35, no. 8, pp. 94–104, August 1997.

[83] H. Xie, P. Narasimhan, R. Yuan, and D. Raychaudhuri, "Data link control protocols for wireless ATM access channels," in *Proceedings of the 4th IEEE International Conference on Universal Personal Communications*, November 1995, pp. 753–757.

[84] J. Mikkonen, C. Corrado, C. Evci, and M. Prögler, "Emerging wireless broadband networks," *IEEE Communications Magazine*, vol. 36, no. 2, pp. 112–117, February 1998.

[85] J. Mikkonen, L. Lehtinen, J. Lahti, and S. Veikkolainen, "A radio access network architecture for IP QoS," in *Proceedings of the 5th International Workshop on Mobile Multimedia Communications*, October 1998, pp. 109–119.

[86] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers," Internet Request For Comments, December 1998, RFC 2474.

[87] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for Differentiated Services," Internet Request For Comments, December 1998, RFC 2475.

[88] P. Almquist, "Type of service in the Internet protocol suite," Internet Request For Comments, July 1992, RFC 1349.

[89] S. Deering and R. Hinden, "Internet protocol version 6 (IPv6) specification," Internet Request For Comments, December 1998, RFC 2460.

[90] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," Internet Request For Comments, September 1997, RFC 2205.

[91] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, January 1991.

[92] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple-node case," *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, pp. 137–150, April 1994.

[93] R. L. Cruz, "A calculus for network delay, part II: Network analysis," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132–141, January 1991.

[94] J. Kurose, "Open issues and challenges in providing quality of service guarantees in high-speed networks," *Computer Communications Review*, vol. 23, no. 1, pp. 6–15, January 1993.

[95] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," Internet Request For Comments, September 1997, RFC 2212.

[96] J. Wroclawski, "Specification of the controlled-load network element service," Internet Request For Comments, September 1997, RFC 2211.

[97] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," *IEEE Network*, vol. 7, no. 5, pp. 8–18, September 1993.

[98] J. Wroclawski, "The use of RSVP with IETF integrated services," Internet Request For Comments, September 1997, RFC 2210.

[99] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," Internet Request For Comments, June 1999, RFC 2598.

[100] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," Internet Request For Comments, June 1999, RFC 2597.

[101] D. D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 362–373, August 1998.

[102] S. Shenker and L. Breslau, "Two issues in reservation establishment," in *Proceedings of the ACM SIGCOMM '95*, October 1995, pp. 14–26.

[103] D. Clark, "Structuring of systems using upcalls," *Operating Systems Review*, vol. 19, no. 5, pp. 171–180, December 1985.

[104] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker, "Agile application-aware adaptation for mobility," *Operating Systems Review*, vol. 31, no. 5, pp. 276–287, December 1997.

[105] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 3, pp. 368–379, April 1990.

[106] D. B. Johnson and D. A. Maltz, "Protocols for adaptive wireless and mobile networking," *IEEE Personal Communications*, vol. 3, no. 1, pp. 34–42, February 1996.

[107] B. D. Noble, M. Price, and 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*, April 1995, pp. 57–66.