A Cross-Layer Perspective on Transport Protocol Performance in Wireless Networks

Stefan Alfredsson

Faculty of Economic Sciences, Communication and IT

Computer Science

DISSERTATION | Karlstad University Studies | 2012:7

A Cross-Layer Perspective on Transport Protocol Performance in Wireless Networks

Stefan Alfredsson

DISSERTATION | Karlstad University Studies | 2012:7

A Cross-Layer Perspective on Transport Protocol Performance in Wireless Networks

Stefan Alfredsson

DISSERTATION

Karlstad University Studies | 2012:7

ISSN 1403-8099

ISBN 978-91-7063-411-6

Distribution:
Karlstad University
Faculty of Economic Sciences, Communication and IT
Department of Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2012

WWW.KAU.SE

A Cross-Layer Perspective on Transport Protocol Performance in Wireless Networks

STEFAN ALFREDSSON
Department of Computer Science, Karlstad University

Abstract

Communication by wireless technologies has seen a tremendous growth in the last decades. Mobile phone technology and wireless broadband solutions are rapidly replacing the last-hop wireline connectivity for telephones and Internet access. Research has, however, shown that Internet traffic can experience a performance degradation over wireless compared to wired networks. The inherent properties of radio communication lead to a higher degree of unreliability, compared to communication by wire or fiber. This can result in an increased amount of transmission errors, packet loss, delay and delay variations, which in turn affect the performance of the main Internet transport protocols TCP and UDP.

This dissertation examines the cross-layer relationship between wireless transmission and the resulting performance on the transport layer. To this end, experimental evaluations of TCP and UDP over a wireless 4G downlink system proposal are performed. The experiment results show, in a holistic scenario, that link-level adaptive modulation, channel prediction, fast persistent link retransmissions, and channel scheduling, enables the transport protocols TCP and UDP to perform well and utilize the wireless link efficiently. Further, a novel approach is proposed where a modified TCP receiver can choose to accept packets that are corrupted by bit errors. Results from network emulation experiments indicate that by accepting and acknowledging even small amounts of corrupted data, a much higher throughput can be maintained compared to standard TCP.

Keywords: Wireless networks, 4G, cross-layer interaction, TCP, UDP, emulation, performance evaluation, optimizations.

Acknowledgements

I wish to thank a number of people that have been important to me during the work on this dissertation. Foremost my advisor, Prof. Anna Brunstrom, has provided excellent support, encouragement and advice on everything from abstract topics down to practical aspects on writing. The same goes for my current co-advisor Prof. Mikael Sternad and the co-advisor for the licentiate thesis, Prof. Tony Ottosson Gadd. My current advisors as well as Lic. Annika Klockar and Dr. Johan Garcia have been co-authors of the published works. Anna Widenius and Dr. Nilo Casimiro Ericsson worked with me early on and built the first version of the WIPEMU network emulator. Dr. Daniel Aronsson was invaluable in the creation of the wireless channel models used in the experiments. Thank you for the nice cooperation.

Thanks to the members of the Wireless IP project, for interesting discussions, workshops and for broadening my knowledge of communication into wireless transmission and related areas.

I acknowledge with gratitude the financial support received by the SSF, KK-stiftelsen, Vinnova, and the European Regional Development Fund provided through the PCC++ graduate school, and the Wireless IP, CMIT, Digicom, Soft Information, and C-BIC projects.

Thanks to the colleagues at the Department of Computer Science, and the members of the distributed systems and communications research group. It has always been a very friendly, open and nice environment to be part of, with interesting discussions.

Thanks to my friends and family. Especially to my wife Christin, for never-ending love, support and encouragement, and to my children Maja and Pontus, for making it all worthwhile.

Karlstad, May 2012

Publications

This dissertation is based on the work presented in the following peer-reviewed papers, technical report and book chapter. The dissertation includes extensions, revisions and additions to the material from these publications.

I Stefan Alfredsson and Anna Brunstrom, "TCP-L: Allowing Bit Errors in Wireless TCP", *Proceedings of the IST Mobile & Wireless Communications Summit*, Aveiro, Portugal, June 2003.

II Stefan Alfredsson and Anna Brunstrom, "Bit Error Transparent Multimedia Transport", Chapter 10 in *Perspectives on Multimedia - Communication, Media and Information Technology*, Wiley 2004.

III Annika Wennström, Stefan Alfredsson, and Anna Brunstrom, "TCP over Wireless Networks", *Technical Report, Karlstad University Studies 2004:21*, Karlstad, Sweden, May 2004.

IV Stefan Alfredsson, Anna Brunstrom, and Mikael Sternad, "A 4G Link Level Emulator for Transport Protocol Evaluation", *Proceedings of the Swedish National Computer Networking Workshop (SNCNW)*, Karlstad, Sweden, November 2004.

V Stefan Alfredsson, "TCP over Wireless Networks: Challenges, Optimizations and Evaluations", *Licentiate Thesis, Karlstad University Studies 2005:13*, Karlstad, Sweden, May 2005.

VI Stefan Alfredsson, Anna Brunstrom, and Mikael Sternad, "Emulation and Validation of a 4G System Proposal", *Proceedings of RadioVetenskap och Kommunikation*, Linköping, Sweden, June 2005.

VII Johan Garcia, Stefan Alfredsson, and Anna Brunstrom, "The Impact of Loss Generation on Emulation-based Protocol Evaluation", *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN)*, Innsbruck, Austria, February 2006.

VIII Stefan Alfredsson, Anna Brunstrom, and Mikael Sternad, "Transport Protocol Performance over 4G Links: Emulation Method-

ology and Results", *Proceedings of the IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Valencia, Spain, September 2006.

IX Stefan Alfredsson, Anna Brunstrom, and Mikael Sternad, "Cross-layer analysis of TCP performance in a 4G system", *Proceedings of the 15th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Croatia, September 2007.

X Stefan Alfredsson, Anna Brunstrom, and Mikael Sternad, "Impact of 4G Wireless Link Configurations on VoIP Network Performance", *Proceedings of the IEEE International Symposium on Wireless Communication Systems (ISWCS)*, Reykjavik, Iceland, October 2008.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In developed countries, mobile cellular penetration has reached saturation, with average penetration now over 100% at end 2010. With more than five billion subscriptions and global population coverage of over 90%, mobile cellular is now de facto *ubiquitous.*

— ITU press release 2011-09-15 of [108]

Communication has been an important part of mankind development, and will be for our future. Communication is about information exchange, and with the introduction of computers and computing devices, there is a need to also communicate human-to-computer, and computer-to-computer. The history of computers and computer communication is short compared to other means of communication, and an exciting topic with lot of developments. To be able to communicate at light speed, sending text, audio and video around the globe, is quite a feat. This dissertation relates to Internet communication carried over wireless networks. Compared to wired networks, wireless networks differ in a few fundamental aspects. This impacts the performance of Internet communications. To give an understanding of the problems involved, this chapter provides an overview of the research area, and related challenges. Further the scope of the dissertation is defined, along with the research methodology and contributions.

1.1 Research area overview

Communication in general happens in a medium that acts as a bearer of the information. Common carriers are electricity in copper wires, light in optic fiber, or electromagnetism in free space. The medium experiences noise from the environment and attenuation with the distance, which makes the signal distorted and increases the risk of incorrect detection by the receiver: the received signal does not represent the same data as the transmitted signal. In a communications system, incorrect data can be detected for example via checksums. When information is transmitted a checksum is calculated and added to the transmission. At reception the checksum is calculated again and compared to the included checksum. If they do not match the receiver knows the received data is corrupt, and can take appropriate action. In some cases data is discarded without further action, and in other cases a retransmission is requested. Further it may be possible to add redundant information, coding, so that information can be corrected, even if parts of it has been received with errors.

Upon this physical communication service, other services are built in a layered fashion. Each layer provides a specific service. For example, in a shared medium such as a broadcast radio system, access to the medium must be controlled. This ensures that communication does not happen simultaneously but is coordinated. Upon this medium access control service, a packet routing service can be constructed. This service can take care of passing packets between nodes and it knows the path between nodes that are indirectly connected. Another service could provide a data stream, in turn using the packet service, for an application.

This is how the Internet is constructed[1]. On top of the physical communication channel, there is a link layer that provides the medium access control. Upon the link layer there is a network layer that provides routing of individual packets between different nodes, using the Internet Protocol (IP) [176]. The transport layer then provides transmission of data upon the packet delivery that IP provides.

[1]See Chapter 2 for detailed descriptions of the protocols and acronyms used here.

The data transmission is available either as a data stream, using the Transmission Control Protocol (TCP) [177], or as data chunks, using the User Datagram Protocol (UDP) [175]. An application can then use TCP or UDP to transmit data to another host without being concerned about how the data gets there, or what type of physical transmission technologies that are being used.

For the layers to provide their services, and because of their independent functionality in relation to each other, transmission problems can be addressed in multiple layers. For example, TCP provides a reliable service to the overlying application, but does not require a reliable service from the underlying network layer. This is managed by retransmitting lost packets. However, other layers may also provide a more or less reliable service. If the link layer detects incorrect data from the physical layer, it may discard that data. If TCP is used, the data will be retransmitted anyway sooner or later according to the protocol design. Alternatively, the link layer may retransmit the incorrect data itself, and thus provide a more reliable service. This in turn adds delay and delay variations, which might interfere with timers in TCP and lead to a performance degradation.

The purpose of this dissertation is to investigate such interactions between the transport, link and physical layers in wireless infrastructure based networks. To further put the research into perspective, the next section presents the research scope and questions considered.

1.2 Research scope and questions

The overall research question of the dissertation is how the performance of Internet access over wireless networks can be improved from the transport and link layer point of view. This is a quite broad question, and it has been necessary to limit the scope of research. Specifically, "Internet access over wireless networks" is limited to Internet access via an infrastructure-based wireless network, where the link layer in question is in the last hop between the base station and the terminal. The wireless network assumes mobility, in the sense that the terminal may physically move in the base station coverage

area. Issues related to handover are not considered in this dissertation. "Internet" relates to TCP/IP and UDP/IP applications. Three specific research questions are considered:

1. *What are the causes of problems in wireless networks compared to wired networks and what transport protocol changes have been suggested for remedy?*

   This question serves as a background of the problem formulation and the dissertation. To understand the underlying problems of wireless vs wired communication, the basics of wireless transmission is reviewed and related to the impact on higher layers. There are many proposals and measurements to optimize transport protocol performance over wireless networks. A number of these, related to the research area, are surveyed and categorized in an effort to further understand the issues involved and possible existing solutions.

2. *What performance can be expected from a more flexible transport service where reliability may be traded for throughput?*

   The surveyed proposals preserve the semantics of full reliability. Inspired by the concepts of partial reliability in UDP-Lite [136], PRTP [37], and TL-TCP [165], a question is raised about the reliability with regards to bit errors in relation to the throughput of TCP. A modification to the receiver TCP ("TCP-L") is devised and implemented to allow for delivery of TCP segments with bit errors to the application. The modification is experimentally evaluated in a network test bed.

3. *How do the lower layers and the transport layer interact in a 4G wireless system depending on link layer configurations, and what are the performance implications of different link layer parameters?*

   Besides optimizations on the transport layer, we want to study the cross-layer interactions between the transport layer and the link/physical layers. The transport layer uses several indicators

such as packet loss and round trip time measurements to decide when to transmit and retransmit segments. When transmitting over a wireless network, packet losses and round trip time variations can have different causes compared to a wired network. Aspects such as link frame scheduling, channel prediction, adaptive modulation and link frame retransmissions in a 4G wireless evaluation system are investigated and related to the resulting performance on the transport layer.

1.3 Research methodology

There are many methods of performing research. Problems can for example be analyzed analytically, where a system is modeled mathematically and solved with equations. With increasingly complex and interconnected systems, this approach may not be suitable depending on the aspects being investigated.

Another method is empirical research, where a model is subjected to specific inputs, and the outputs are studied. This is especially useful in complex environments where an analytical model may be too abstract. Simulation is one common method used for such research. A simulated system implements the model entirely in software. This means that the studied system needs to be re-implemented (or, if software source code is available, adapted from that) within the simulation tool. This could be a drawback if the implementation does not exactly behave as the system being modeled. For example, source code of proprietary applications or protocol specifications may not be available. On the advantageous side, a simulation can usually be run faster than real-time and be performed in parallel, compared to measurements on a real (non-simulated) system. Simulation can also be used to evaluate systems that do not yet exist.

A compromise between experiments with a real and a simulated systems is emulation. Here, parts of the system are simulated, and interact with other parts of a real system. Relating this to network research, with emulation real applications and protocols can be used in end nodes. Parts of the network can be emulated to study the

effect of specific settings on the resulting performance on other parts of the system.

The research methodology in this dissertation belongs to empirical research in the form of experimental network emulation. Conceived ideas are implemented in network test beds. Experiments are then performed in these test beds. Measured output from these experiments are compared with the output from experiments with a baseline (standard) configuration. From such comparisons conclusions are drawn as to what effect a certain modification or implementation has. It is often impossible, or impractical at best, to test all combinations of parameter settings and metrics, as these grow exponentially for each new parameter. The scope of the evaluated parameters has therefore been limited by necessity, but is believed to capture relevant aspects of the investigated protocols and systems. However, multivariate analysis [193] can be used to reduce the parameter space and could be considered to be used in future work.

To strengthen the validity of the experiment results, the output from experiments in the test beds have been compared to the theoretically achievable output. The comparison confirms that the emulated system delivers the expected results over a set of known parameters.

To increase the generalizability of the experiment results, each experiment with the WIPEMU emulator (described later) has been replicated with three different channel models, with 30 different channel realizations each to provide statistical validity with a mean value and a 95% confidence interval. Experiments with TCP-L (also described later) also used 30 replications, but used a simplified emulation and channel model employed before the development of WIPEMU.

1.4 Main contributions

The main contributions of the dissertation are outlined below. As mentioned in the publication list in the preamble, the dissertation is based on previously published works by the author. These are cited below to reference the original publication venue and time.

First, cross-layer interactions between the physical, link and trans-

port layers in a wireless 4G evaluation system are investigated [9–11]. General mechanisms for a 4G system such as link layer retransmission persistency and delay, fixed and adaptive modulation, channel prediction and scheduling, in combination with different application scenarios are studied. The results indicate that link layer design gains can be achieved at the transport layer. There are previous analytical and simulation results that confirm this from a channel perspective in isolation, but the studies presented here are believed to be the first, at the time of initial publication, to experimentally verify this in a holistic setting. Further, bulk data transfer using TCP, as well as real-time VoIP traffic using UDP, are shown to perform well with the same parameter configurations.

Second, a TCP receiver modification ("TCP-L") is developed that allows an application to choose a more flexible transport service [6,39] where reliability can be traded for throughput. An application can, by accepting bit errors in the data stream, achieve a higher throughput compared to using regular TCP in cases where residual bit errors are the cause of packet loss at the receiver. Because TCP congestion control multiplicatively decreases the sending rate in response to packet loss, using TCP-L can provide a significant performance improvement since unnecessary invocations of congestion control are avoided. This is mainly useful when link layer retransmissions are not used, or if used, have a long retransmission time compared to the total end-to-end delay.

Third, a survey of TCP optimizations for wireless networks is done [240]. The optimizations are categorized into end-to-end, link layer, explicit notification, and split connection approaches. This provides a background and presents related work for the optimizations and evaluations in the dissertation.

Fourth, a significant amount of emulation software is developed to support the research outlined in the first and second contributions. The network emulator WIPEMU [7,8] is constructed to emulate the wireless downlink in a 4G evaluation system. This development effort was needed to be able to perform the experiments on the Wireless IP system, since no other suitable emulation system was available to us at the time. WIPEMU was used to provide the primary research

results. For the experiments on TCP-L, a modification to Dummynet [42, 188] was implemented to insert bit errors into passing packets in a deterministic and controlled way. This initial work was continued by other colleagues to form the KauNet [81, 84] network emulator.

1.5 Outline

The remainder of the dissertation is organized as follows. Chapter 2 gives a background on data communication with a focus on wireless communication. Differences between wired and wireless communication are outlined. Further the chapter provides an introduction to wireless network architectures and their characteristics in relation to the topic of this dissertation. The chapter concludes with a description of the Internet protocols and issues of TCP over wireless. Chapter 3 presents related work and begins with a survey of TCP and link layer optimizations for wireless networks. Further the chapter presents performance evaluations of deployed networks, and discusses network emulation. Chapter 4 conveys the idea and motivation for a more flexible transport service, where lower reliability can be traded for higher throughput. A TCP modification called TCP-L is developed, where TCP packets with bit errors can be accepted instead of retransmitted. The chapter presents the motivation, implementation, experiments and results with TCP-L. Chapter 5 introduces the Wireless IP evaluation system. The system properties is described, as well as the implementation and validation of the emulator WIPEMU. Following this, WIPEMU is used in Chapter 6 to evaluate the effect of a number of link layer parameters on the transport layer. Real TCP and UDP applications are used to communicate through the emulated system, while performance data is captured. The chapter analyzes and concludes the cross-layer performance impacts of the link layer parameters on the transport layer. Finally, Chapter 7 concludes the dissertation and presents ideas for future work.

Chapter 2

Communication background

You see, wire telegraph is a kind of a very, very long cat. You pull his tail in New York and his head is meowing in Los Angeles. Do you understand this? And radio operates exactly the same way: you send signals here, they receive them there. The only difference is that there is no cat.

— Albert Einstein

This chapter provides a background and an overview of the fundamentals of communication. The focus is on wireless communication and Internet traffic. To explain the issues of TCP over wireless, the chapter presents a brief history of communication, properties of wireless networks, and the function of the Internet protocols. The chapter concludes with a discussion of the challenges of using TCP in wireless networks.

2.1 Brief communication history

Communication in its simplest form consists of transmitting information from one entity to another, via a medium. In the case of computers, communication is often performed via metallic mediums, such as copper on a circuit board, coaxial cable, or twisted pair wire,

popular in the pervasive Ethernet technology. In the case of metallic medium, electrons are used to carry the signals. Glass fiber is also commonly used for communication, but here photons (light) are used instead of electrons. All these mediums require some form of mechanical connection between the communication end-points. Starting with the inventions of Tesla and Marconi some 120 years ago, there is the possibility to communicate without using wires, as electromagnetic waves do not require any wired medium to be present. This allows for a great amount of communication mobility, as phones and computers can communicate without being bound by a wire.

During the last decade, wireless communication equipment has become readily available for use with personal computers. Wireless local area networking is commonly done with the IEEE 802.11 (WiFi) standard [227]. Wider area coverage is made possible by utilizing the existing mobile phone infrastructure for data transmission in the form of for example GPRS [28, 91]. The integration between packet data and voice communications is further evolved with the deployment of 3G/UMTS [221]. This system is specifically designed to carry packet data, video and voice communications with much higher capacity than previous wide area coverage networks. 3GPP LTE (discussed later) is currently being deployed with even more focus on packet data, and provides an order of magnitude higher capacity than 3G.

Parallel to this development is the expansion of the Internet, which has seen an explosive growth since the introduction of the World Wide Web in the early 1990s. It seems like a natural evolution that the Internet should be accessible wirelessly, just like the mobile phone is superseding the fixed line phone [108]. Recent studies [108] find that mobile cellular penetration is above 100% in developed countries, with a global population coverage of over 90%. Likewise more and more Internet users can be expected to use a wireless instead of a wired connection. An example of this is the marketing in, for example, Sweden of mobile broadband that uses USB 3G modems plugged directly into the computer. It competes directly with fixed line ADSL broadband in terms of (advertised) throughput of up to 7.2 Mbit/s.

2.2 Layering models

Computer communication is often structured in a layering model [223]. Each layer performs a specific task. Layers are stacked upon each other. A layer provides services to a layer above, and requests services from the layer below. As long as the services and interfaces between the layers are intact, they can be exchanged while the system as a whole continues to function. For example, a web browser application will work the same when the computer is connected via Ethernet cable as well as when a wireless connection is being used. This is because the layer handling the medium access provides the same service to the layers above, independently of a wired or wireless communication medium.

There are two major layering models. The ISO *Open Systems Interconnection* (OSI) model [106, 223] define seven layers: application, presentation, session, transport, network, data link and the physical layer. For the Internet, a more compact layering model is used, defined by the IETF. It defines [35, 213, 223] four layers: application, transport, network, and link/network access layer. Compared to the OSI model, the application layer is assumed to encompass also the presentation and session handling. An overview of the two layering models is shown in Figure 2.1. The OSI reference model has found mainly academic use[1], while the IETF TCP/IP model is the de-facto protocol suite used on the Internet. We therefore focus the discussion on the latter.

The *application* layer is the communication end-point, and is usually an interface to the user or operator. This can for example be a web browser, web server, email client, or email server.

The application uses the services of the *transport* layer to communicate with another end point. The transport layer can provide a reliable data stream transmission over an unreliable packet network. In the IETF model the TCP protocol provides this reliable data stream transmission. There is also a packet based unreliable transport protocol, UDP, as well as other more specialized transport

[1] Reasoning about this is in [223, Ch 1].

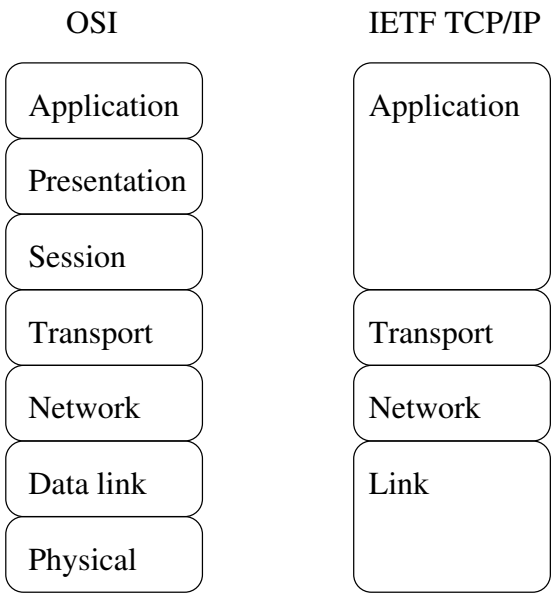


Figure 2.1: The 7 layer OSI reference model and the IETF TCP/IP model.

protocols. TCP, UDP are described in more detail later in the chapter.

The transport layer divides the data stream into packets, and uses the *network* layer to deliver these to the communication end point. At the receiving end, the transport layer is responsible for collecting the packets from the network layer and for reassembling these into a data stream to the correct application. The network layer provides routing of data packets from one end to another. It keeps a routing table that instructs which node should be used for a given destination, and relays incoming packets there. Eventually, the packets reach the final destination and are delivered to the transport layer of the receiving node.

The *link* layer provides direct communication between two nodes with the connection facility that is available between them. It transmits frames of information, where a frame may be smaller than the packets it receives from the network layer. The link layer performs medium access control to know when frames can be transmitted. It may perform retransmissions and error correction in case of lost or

corrupted frames. At the receiving end, the frames are assembled and delivered to the network layer for further processing. Included in the IETF link layer is the *physical* layer which is responsible for the actual transmission of information between the two nodes. How it works is dependent upon the physical medium utilized. Nodes may be connected with a metallic wire which can then use electricity for transmission, or use fiber optic cables with photons, or use antennas and electromagnetism, as has been mentioned earlier. In the next section this last technique is described in more detail.

Traditionally each layer has access to the immediate layer above and below via standard interfaces. The application can access the transport layer, the transport layer can access the application and the network layer, and so on. With *cross-layering* control mechanisms the access is provided in a more flexible way, either between immediate layers or between layers that are otherwise isolated. This access may include control or information exchange, outside of the layered programming interfaces. Cross-layer interaction studies investigates effects that exist due to cross-layer interaction effects. For example both the transport layer and the link layer may or may not perform retransmissions, which could affect the behavior and performance of the other layer. One of the purposes of this dissertation is to investigate cross-layer interactions between the transport and link layer.

2.3 Wireless networks

We now focus the attention to the properties of wireless networks from a bottom-up perspective. The physical and link layers in the layering models are discussed. How information is transmitted on the physical layer, solutions to provide multiple access, network architectures and the development and standardization processes are discussed below.

2.3.1 Wireless communication

Wireless communication in the context of this dissertation is assumed to be based on electromagnetic transmission, with transmitting and receiving antennas. Some of the important aspects of wireless communication is how to convey the actual information, what capacity can be achieved, how signal quality is affected and how wireless transmission differs from transmission in a wired medium. This is discussed in the following subsections.

Modulation

To facilitate data transmission in a medium on the physical layer, a carrier signal is often used (Figure 2.2). This carrier is modulated, or modified, in certain ways according to the data being transmitted. Common modulation techniques are amplitude, phase and frequency modulation [90]. For example, common broadcast radio transmissions used amplitude modulation for a long time, known as AM radio. This modulation scheme uses a fixed carrier frequency, and the amplitude of the signal indicates the information transmitted. For digital/binary transmission, the "full" amplitude may represent a `1`[2], and half of that amplitude may represent a `0`. Amplitude modulation is illustrated in Figure 2.4, where the carrier has been modulated with the signal in Figure 2.3.

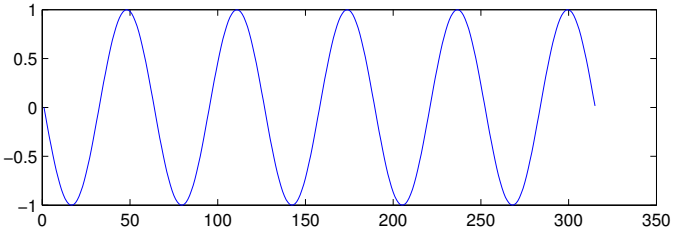


Figure 2.2: An example carrier signal

When phase modulation is used, the phase of the carrier signal is shifted. With our example, a `1` may be represented with a phase

[2]All numbers are in base-2 (binary) representation in this context.

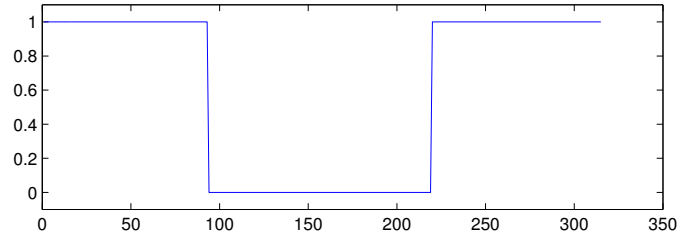


Figure 2.3: A data signal

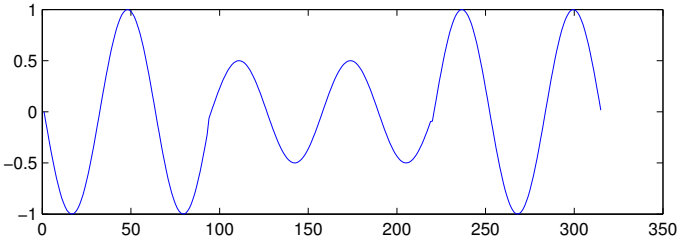


Figure 2.4: Amplitude modulation

shift of zero degrees, while a `0` may be represented by a 180 degree phase shift, as illustrated in Figure 2.5.

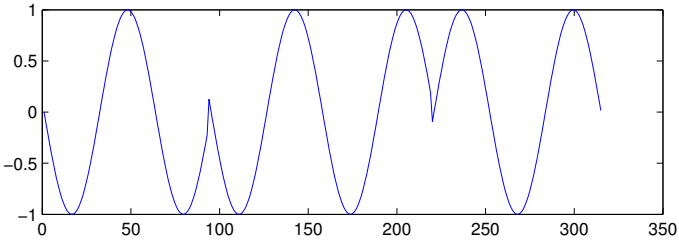


Figure 2.5: Phase shift modulation

The third way of modulation is frequency modulation, which is widely used in FM-radio broadcasting. Here, the frequency of the carrier signal is varied according to the data that is to be transmitted. For example, a `1` may be represented by a frequency of 800 Hz, while a `0` may be represented by a frequency of 400 Hz, as illustrated by

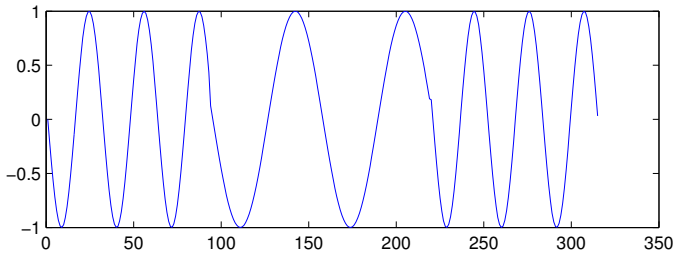


Figure 2.6: Frequency modulation

Figure 2.6. These examples assume transmission of only `0`'s and `1`'s, by using two distinct modulation levels. By adding more levels, for example 25%, 50%, 75% and 100% of the maximum amplitude, more bits can be represented per time unit (i.e., per transmitted *symbol*). This can be increased even further, from 4 to 8, 16, or 32 levels and so on. The limit for the number of modulation levels (or modulation order) is dependent on how clear the signal is compared to the noise[3], or the *Signal to Noise Ratio* (SNR). For example, 25% (of the maximum amplitude) may represent `00` and 50% may represent `01`. The sender transmits at 50%, but on the path to the receiver the signal may have become attenuated to 33%. This symbol may then be erroneously demodulated as `00` by the receiver, since the distance is closer to that decoding alternative. The higher the signal to noise ratio, the more levels can be used, and thus a higher capacity is obtained.

To further improve capacity, different modulation techniques can be combined. For example, amplitude and phase modulation is commonly combined. By using two modulation levels for both amplitude and phase, four different combinations (symbols) are created, while still being as robust to noise as each of the modulations themselves. The combination of amplitude and phase modulation is commonly referred to as *Quadrature Amplitude Modulation* (QAM).

[3]The noise may also include interference from the sender itself, or other sources.

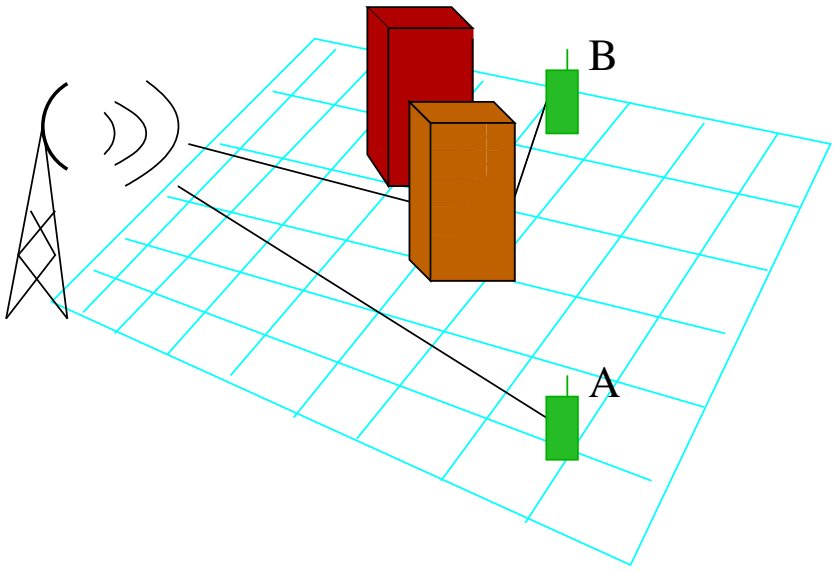


Figure 2.7: Illustration of a line-of-sight transmission to terminal A. Terminal B does not have line-of-sight to the base station and only receives reflections of the signal.

Fading

An important characteristic of wireless communication is fading [90]. Fading in this context relates to channel power variations at the receiver. There are a number of reasons for these variations. Intuitively, a signal is weakened by the distance between the transmitter and receiver, as the transmitted energy spreads out. The amount of weakening depends on the dissipation model, as discussed later. This type of fading occurs on longer time scales. Another type of fading occurs on short time scales, for example due to noise, reflection, diffraction, refraction and scattering. Since the medium can not be shielded like a wire, signals in the same frequency spectrum can interfere with each other and increase the noise component in the receiver. Considering an omni-directional or semi-directional antenna, the radio waves that reach the receiver may arrive directly from the transmitter. The communications is said to be in *line-of-sight* (LOS). Often there are additional components in the received signal, caused

by waves that have bounced off buildings, off the ground, or off other objects in the path, like echos of the original signal. These waves then contribute to noise, as parts of the signals are delayed and all these signals are combined at the point of reception. In urban terrain, line of sight may be rare (*non-line-of-sight* (NLOS)) because of buildings and other objects in the path from the transmitter to the receiver. See Figure 2.7 for an illustration where a terminal A has line-of-sight to the base station while the signal to terminal B is blocked by objects, and B is reached by reflections of the signal. This complicates the detection further, as all received signals are reflections from other objects. This is a characteristic of radio transmission that is non-directional. For example directional laser or microwave point-to-point wireless transmission, does not experience such problems.

Channel capacity and energy distribution

Of interest is the attainable channel capacity. This depends on a number of factors. A central part is the amount of bandwidth available, and the signal quality (signal-to-noise). The higher the bandwidth and signal quality, the more information can be transmitted during the same time interval. The theoretical upper limit C bits/s is determined by the Shannon-Hartley Limit, Eq. 2.1 [198].

$$C = BW * log_2\left(1 + \frac{S}{N}\right) \tag{2.1}$$

Here, BW is the bandwidth in Hz, S the signal power and N the noise power. From the formula, it is seen that the capacity is directly proportional to the bandwidth, and logarithmically proportional to the signal to noise ratio. To improve capacity, it would therefore be more efficient to increase the bandwidth than to increase the transmission power. By deriving from the capacity equation, [55, Ch 2.1.1] concludes that "to make efficient use of the available received power, or, in the general case, the available signal-to-noise ratio, the transmission bandwidth should at least be of the same order as the data rates to be provided". Increasing the transmission bandwidth is thus

a key factor to providing high-capacity communication services.

With regards to the capacity versus distance, a higher transmission power level means that the signal will reach further. Alternatively, if the distance is the same, the increase in power will provide a better signal to noise ratio and thus higher capacity.

The amount of received power depends of course on the power output from the system, but also on the dissipation model. An omnidirectional dissipation, such as standard broadcast radio, spreads the energy in all directions. A directional dissipation, with beam forming or a laser, concentrate the energy towards a given direction. This means that for the same amount of energy, a higher capacity will be available. It will, however, be at the expense of the number of possible receivers since the area where the transmission is directed is reduced.

For infrastructure based mobile networks, a common configuration is to have sectored antennas. This way, a cell tower can have 360 degree coverage, but achieve greater capacity than with an omnidirectional antenna. Since the sectors are smaller, such as 120 or 60 degrees, different frequency bands can be used in each sector, to provide a separation of users and increase capacity compared to an omni-directional antenna. This is also called *space-division multiple access* (SDMA).

An interesting topic in antenna technology is *multiple input, multiple output* (MIMO). A MIMO device then has multiple antennas. This means that a received signal can be received with a short time delay and via different paths by the different antennas. With advanced signal processing, the original signal can then be extracted with a better signal to noise ratio, compared to reception with a single antenna.

Coding

One method to alleviate the effects of channel distortions is to apply coding [90]. Coding consists of transmitting additional information, such that errors can be detected and possibly also corrected. The information with coding added is called a code word.

A simple error detecting code is the *parity bit*. The parity bit is used to form an even ("even parity") or odd ("odd parity") number of `1` bits in the code word. If, after reception, the parity does not match, this indicates that there is an error in the code word. Parity is able to detect single (or an odd number of) bit errors, but fails for double (or an even number of) bit errors since they produce the same parity. A more advanced error detecting code is the *cyclic redundancy check* (CRC), that calculates a checksum over blocks of data, and is able to detect more errors compared to the parity check. The CRC-32 checksum is commonly used for example in Ethernet frames, image compression such as PNG and file compression programs such as gzip.

If an error has been detected, the error may be corrected by using an error correcting code. The error correcting code contains redundant information that can be used to correct single or multiple bit errors, depending on the type of code in use and the amount of redundant information. A simple example is to transmit a code word twice. If the first code word fails a parity check, it can be compared to the second code word bit by bit to find the bit or bits that differ. There are more advanced techniques, such as Hamming, Reed-Solomon, Convolutional and Turbo codes [90].

Wireless and wireline differences

Comparing wireless to wireline transmission, the wire is a guided and often shielded medium. This means that either much less power needs to be used, or conversely, that the communication distance can be increased. A shielded medium blocks external interference, and also works the other way - transmissions in a wire does not normally[4] *cause* interference. This means a much greater freedom in choosing frequencies to use, compared to the heavily regulated radio spectrum for wireless transmission. Thus, wireline transmission allows for a much higher bandwidth, non-shared, with a lower error probability, compared to wireless transmission.

[4]There is always a magnetic field around a wire with electric current. Also, the wire may act as an antenna. For our discussion these issues are not considered.

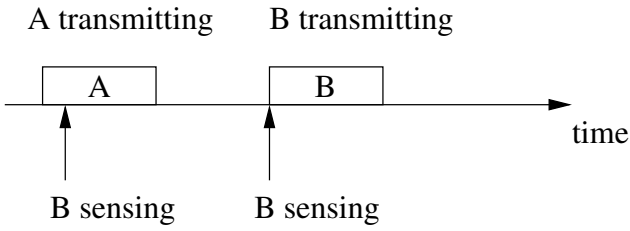


Figure 2.8: Illustration of Carrier Sense Multiple Access. Terminal B wants to transmit, and senses the channel. Since a transmission of A is detected, B waits for a time interval and senses again. Now the channel is available and B can transmit.

2.3.2 Multiple access

To have an efficient utilization of a communication channel, it is desirable to allow multiple users access. There are a number of ways in which this can be achieved. A common scheme is *Carrier Sense Multiple Access* (CSMA) [223]. Here the terminal that is about to transmit first tries to determine if the medium is available or not. If the channel is occupied, the terminal waits a period of time and then sense the carrier again, until the channel is free. A drawback with CSMA is that two terminals may sense a free channel at the same time. For example, if the signal from the first terminal has not yet reached the second terminal, the second terminal will also start to transmit. This may cause a collision at a receiver which can not decode any of the signals. A solution to this is to use additional signaling. The sender sends out a request-to-send, and receives a clear-to-send message. The other terminal is then aware of a foregoing transmission[5]. Due to the opportunistic nature of this multiple access scheme, it is often used in ad-hoc networking where no infrastructure is available to coordinate transmissions. An illustration of CSMA is shown in Figure 2.8. CSMA can also be illustrated with a simple analogy. Imagine a group of people having a discussion. Before a person speaks out, he or she listens if anyone else is speaking.

[5]There are also problems with this approach, such as the hidden node problem [166].

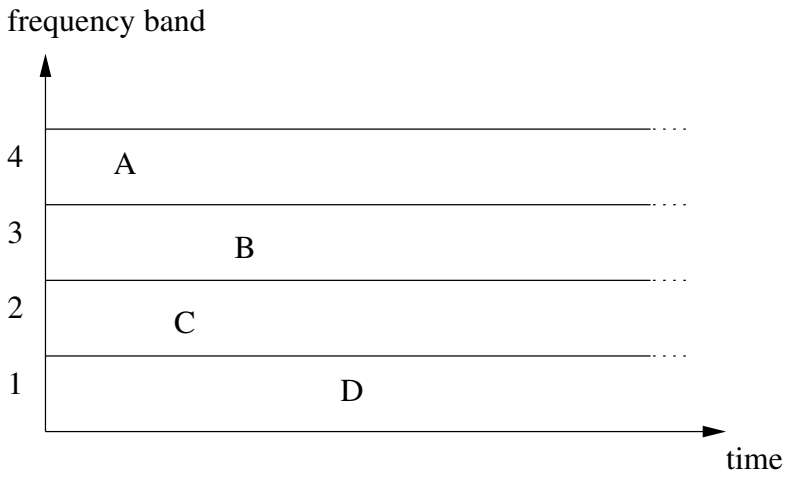


Figure 2.9: Illustration of Frequency Division Multiple Access. Terminals A, B, C and D are exclusively assigned different frequencies for transmission, which can be used without time synchronization between terminals.

With *Frequency Division Multiple Access* (FDMA) the channel is divided into several frequency bands. A terminal is then allocated a frequency band to use at its own discretion. Because of this partitioning, multiple terminals can communicate at the same time without interfering with each other. There are also some drawbacks with FDMA. Since the available channel is divided, there is less capacity per terminal compared to if one terminal utilized the whole channel. Also, if a terminal is idle and does not transmit or receive, the frequency band is still occupied and can not be utilized by another terminal. Figure 2.9 shows a FDMA partitioned channel. To use the group discussion analogy, persons that communicate speak with a certain voice pitch, that can be heard by the other persons.

Another technique is *Time Division Multiple Access* (TDMA). Here, the time is partitioned into time slots which can be allocated to different terminals. Like FDMA, terminals can then utilize the channel at their discretion, but only for a determined period of time. The time slot is then recurring, depending on the total amount of slots and the slot length. An example of this is shown in Figure 2.10. An analogy here would be people taking turn when speaking, such

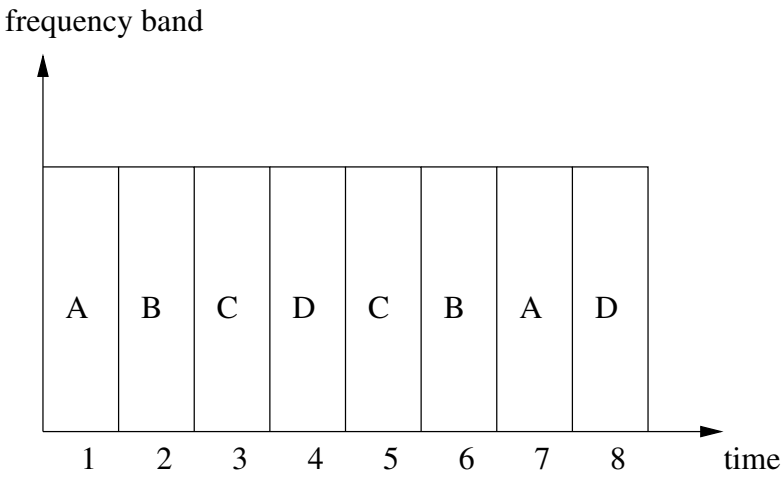


Figure 2.10: Illustration of Time Division Multiple Access. Terminals A, B, C, D are assigned individual time slots during which the whole frequency band may be used. Transmission is only allowed within allocated time slots.

as talking for 10 seconds each and then being quiet until everybody else has spoken.

The mentioned multiple access schemes each requires separation between users in one way or another. Either by checking that no one else was communicating, by having a separate frequency band, or by having separate time slots. A multiple access scheme that does allow overlapping communication is *Code Division Multiple Access* (CDMA) [90]. Here users communicate in the same frequency bands at the same time. Transmissions are encoded with carefully chosen codes. The receiver can distinguish transmissions with a particular code, even though other codes are sent at the same time. Figure 2.11 illustrates CDMA. Using the group discussion analogy, everyone would talk at the same time and with the same volume and pitch, but using different languages. Those speaking English would be able to hear each other, and the other languages would be heard as background noise.

A fourth common multiple access technology is *Orthogonal Frequency Division Multiple Access* (OFDMA) [90]. In single-access OFDM, the transmission is modulated over a number of sub-carriers

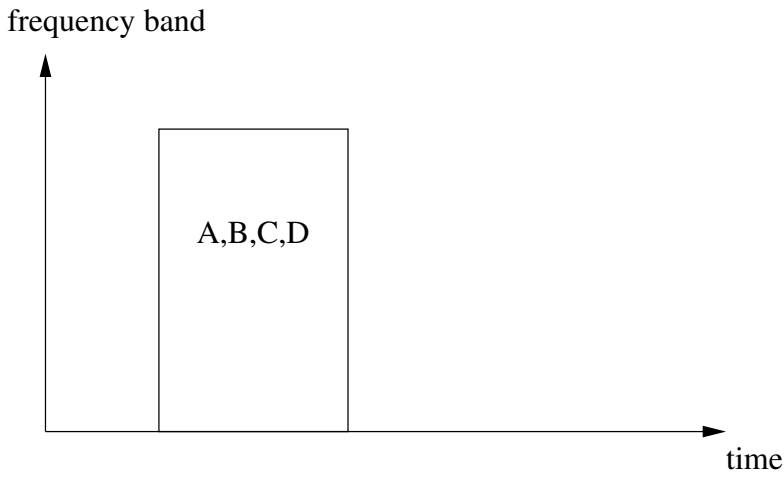


Figure 2.11: Illustration of Code Division Multiple Access. Terminals A, B, C and D all transmit at the same time and at the same frequency.

over the available frequency band. OFDMA works by grouping some of the sub-carriers to sub-channels. These sub-channels are then allocated to different users by also using FDMA and/or TDMA (as is further explained in Chapter 5).

2.3.3 Wireless network architectures

A key property of a wireless network technology is the architecture. The architecture influences to a large degree which technologies that are used in the network. For example, the network may or may not use an infrastructure. In a network without infrastructure, transmissions are often uncoordinated and nodes use carrier sensing to know when to transmit. In an infrastructure based network which uses time or frequency division, the infrastructure decides in which time slots or frequency bands to transmit.

The network architecture also defines the physical and logical reach of communications. With a network that has an infrastructure of base stations, a wider coverage can be provided than with a single base station. This can also be achieved without an infrastructure such as is done with mobile ad-hoc networks. However, the

reliability and performance of such a network is highly dependent on the participating nodes. With an infrastructure, connectivity to other networks is usually available. The routing is clearly defined, and the exit route is normally in the base station. This can be compared to an ad-hoc network where one or more of the nodes may have connectivity to other networks, and an indefinite number of nodes must be traversed until an exit node is reached.

This dissertation is focused on infrastructure based mobile wireless networks. The section continues with a discussion about the meaning of "mobile", and properties of wireless "local" and "wide" area networks.

Wireless and mobility

There are many technologies that can be labeled wireless, but differ in terms of mobility. The concept of mobility is here used in the sense of physical movement of a terminal while maintaining a network connection. On one side of the mobility scale, there are stationary wireless nodes, which do not move at all. On the other side of the scale there are nodes moving at vehicular speed or higher. For example a cell phone user in a car or train could be moving at 50 to 150 km/h.

An example of highly stationary wireless communication is microwave or laser links. These provide directional point-to-point transmission, usually with very high[6] bandwidth. Mobility in the traditional sense (i.e., on a short time scale) does not exist, unless the antennas are unmounted, moved, mounted and re-calibrated.

See Figure 2.12 for an illustration. Moving along the scale, we find personal area networks with a range of tens of meters. Further we find the ubiquitous WiFi/WLAN technology that uses an access point or base station and allow for mobility within the range of the radio transmission, up to hundreds of meters. The mobility range can be extended by connecting multiple base stations sharing the same network id and possible encryption keys. The client can then do a

[6]In the range of hundreds of megabit/s to gigabit/s capacity.

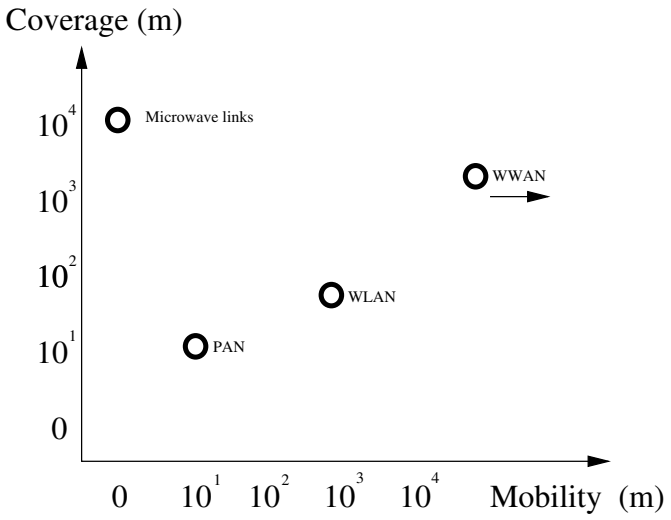


Figure 2.12: Illustration of coverage vs mobility of stationary wireless, personal area, wireless local area and wireless wide area networking.

handover between different access points, to get continuous service while maintaining the network address.

Whereas WiFi mobility is often limited to a building or small area, a cellular phone system allows for mobility over a large area. Radio base stations are deployed to cover most of the population, up to tens of kilometers each, and allow for seamless handover between them.

Wireless Local Area Networks

Wireless *local* area networks provide services in small areas, typically ranging from ten up to several hundred meters in radius. For an infrastructure based local area network, it is constructed with one or more base stations with which the nodes communicate. The most common wireless local area networks for packet data today are built upon the IEEE 802.11 (WiFi) [227] standard. Another common local area network technology is Bluetooth [226], popular for short range communication between computers, cell phones and other devices. The range of Bluetooth is typically shorter than WiFi, and is there-

fore called *personal* area network (PAN). This also reflects from the usage, where Bluetooth is mainly used for communication between personal devices, compared to WiFi which typically is provided by another party and shared by different users.

IEEE 802.11

The original 802.11 standard [227] allows for a rate of 2 Mbit/s. It uses a direct-sequence spread spectrum transmission technology in the unlicensed 2.4 GHz frequency spectrum. This means that other devices that operate in the same frequency may cause interference, such as microwave ovens. There were also concerns when Bluetooth was introduced that these two technologies would interfere, as Bluetooth also transmit in the 2.4 GHz spectrum. The spread spectrum transmission technology that is used is, however, beneficial in the presence of interference. The signal is spread out over a larger frequency band, compared to if a single carrier would have been used. This means that it is more resilient if other equipment is interfering at specific frequencies. Interference may happen with other 802.11 devices. To alleviate this, the medium access control protocol employs the carrier sense algorithm discussed previously.

An IEEE 802.11 network is identified by a *Service Set ID* (SSID), also called network name. By having multiple base stations with the same SSID to provide an extended service set, clients can associate with any one of them. If the base stations are within coverage of each other, a client can have seamless access while moving around. When one base station goes out of reach, the client can associate with another with better signal quality. If all base stations are connected to the same network backbone, the client can keep its IP address during this movement.

IEEE 802.11 b/g/a/n

The original standard has been extended a number of times, while keeping backwards compatibility. 802.11b increased the transmission rate to 11 Mbit/s. In 802.11g and 802.11a, the rate is increased further to 54 Mbit/s by using OFDM modulation. 802.11a also operates in the 5 GHz band, which allows for less interference with legacy

802.11 standards and other equipment in the 2.4 GHz band [223].

The latest revision is 802.11n, where the biggest change is the introduction of support for multiple antennas, MIMO. As mentioned earlier, with MIMO it is possible to combine the signal received at all the antennas to get a better signal quality, and implicitly higher capacity [103].

Bluetooth

Bluetooth is another common wireless network technology, with a reach of tens of meters. It is called *personal* area network, due to its intended use. The original purpose was to replace cables connecting personal devices. Examples include phone headsets, mobile phone cables, and computers. The headset can be connected, or *paired*, to the phone and audio is then diverted to the headset like a cable had been used.

Bluetooth operates with direct-sequence spread spectrum and direct-sequence frequency hopping in the 2.4 GHz band. In comparison to 802.11, Bluetooth does not support roaming in the same sense. A Bluetooth network (*piconet*) is built up by one node acting as a master, and then there can be up to seven slaves connected to the master. Nodes can also join multiple piconets to form *scatternets*, which can cover a larger area and more devices than a single piconet.

Wireless Wide Area Networks

Wireless *wide* area networks provide services over much larger areas. The coverage ranges from kilometers in radius and upwards. The reach is limited by the number of base stations deployed. Deployment is more formal than for local area networks, where anyone can install a base station. Wide area networks are deployed by large companies, due to the cost of equipment and investment requirements. Due to its reach, the frequencies used are regulated by the government to ensure that they do not interfere with other communication equipment. The base stations are carefully placed into cells to provide good coverage and efficient frequency reuse. In areas with high mobile terminal density, the cells are smaller to allow for more cells in the area. On

the opposite, in areas with less terminals the cell size can be larger to save costs for base stations.

The most widely used and publicly accessible wide area networks are the cellular mobile communication systems (see [55, 145, 196] for further details of the systems described below). These can be grouped into a number of generations, based on the kind of technology used. The first generation, 1G, was built upon analog transmission and provided analog voice services in the early 1980s. In the United States, this technology was known as *Advanced Mobile Phone System* (AMPS), while in Europe NMT-450 was used. As the interest for data communication increased, there was a need to also provide data transmission on a wide scale. For wired phones, modems were used to achieve data communication over analog phone lines. The modem modulated the data into tones, sent over a phone connection, and the tones were demodulated at the receiver. The same method could be used with the mobile phones. A more efficient way is to send packet data directly in the network. An extension to AMPS, *Cellular Digital Packet Data* (CDPD), was created for this purpose. A CDPD enabled device listened on the AMPS traffic, and when a channel was not used for voice communication CDPD used the channel for data transmission.

The second generation mobile systems, 2G, are based on digital technology. Instead of analog transmission, the voice is digitized and sent digitally. Initially, the *Digital AMPS* (D-AMPS) system was used in the United States, while in Europe the *Global System for Mobile communications* (GSM) system is used. Nowadays GSM is available world-wide. Because of the requirements of packet data transmissions, this is integrated into these systems. In GSM, this is handled by the *General Packet Radio Service* (GPRS). GSM is built on TDMA (discussed earlier), with eight time slots per channel. GPRS then uses one or more of these time slots for data transmission. This means that the GPRS throughput can vary depending on the amount of free slots, but can reach up to 80 kbit/s. An evolution of GPRS, EDGE (described as 2.5G), uses higher modulation along with incremental redundancy error correction, and triples the bit rate of GPRS to 244 kbit/s.

The third generation mobile system, 3G, is also based on digital transmission. The big difference of 3G over 2G is the use of CDMA for the radio interface. This means that more users can be accommodated, and there is a soft quality degradation if the network becomes overloaded. In the United States, the cdma2000/EV-DO network is based upon this technology. In Europe, *Universal Mobile Telecommunications System* (UMTS) is used. UMTS is based on GSM, and allows reuse of some central network components such as the home location register and mobile switching center. UMTS is evolving and is specified in numbered releases.

With 3G systems, even more focus is put on packet data transmission, and is the first commercial releases were designed for speeds up to 384 kbit/s (UMTS Rel. 99). UMTS Release 5 includes the high speed downlink packet access extension, HSDPA, to achieve up to 14 Mbit/s downlink speeds, while UMTS Release 6 provides a speed increase to 5 Mbit/s for the uplink (HSUPA).

Currently "4G" systems are being deployed and sold commercially [225], while industry and academic research continues to further increase capacity, increase coverage and reduce latency of future mobile networks. The next section discuss some of the related development, research and standardization efforts in more detail.

2.3.4 Development and standardization of 4G systems

As the third generation mobile systems were being deployed around year 2000, research on future mobile systems started to form. These were named Beyond 3G (B3G) or 4G, although at the time there was no agreed definition of the term. As of December 2010, the ITU has pronounced that the IMT-Advanced systems are considered "4G", for example in the following quote from a press release[7].

[7]ITU press release 2010-12-06, "ITU World Radiocommunication Seminar highlights future communication technologies", `http://www.itu.int/net/pressoffice/press_releases/2010/48.aspx`

> Following a detailed evaluation against stringent technical and operational criteria, ITU has determined that "LTE-Advanced" and "WirelessMAN-Advanced" should be accorded the official designation of IMT-Advanced. As the most advanced technologies currently defined for global wireless mobile broadband communications, IMT-Advanced is considered as "4G", although it is recognized that this term, while undefined, may also be applied to the forerunners of these technologies, LTE and WiMAX, and to other evolved 3G technologies providing a substantial level of improvement in performance and capabilities with respect to the initial third generation systems now deployed. The detailed specifications of the IMT-Advanced technologies will be provided in a new ITU-R Recommendation expected in early 2012.

Development and research of systems are carried out by industry and academia, in conjunction with standard bodies to produce interoperability.

Mobile system research and development

Mobile systems research and development is driven to meet operator and customer demands in terms of higher capacity, larger coverage, and other factors. Whereas this can result in vendor specific systems, the success of inter-operable mobile systems such as GSM and UMTS means that future systems should also inter-operate. This allow customers to use the same handset everywhere, compared to having to use different handsets because of different technologies. Operators also benefit from lower cost of network equipment due to economy of scale and increased numbers of roaming users. To a large extent GSM and UMTS fulfill this purpose, but as capacity demands increase new systems need to be developed. This is thus carried out in cooperation of involved parties in the telecommunication industry and academia, in various constellations.

The *Wireless World Research Forum* (WWRF) [243] is such a constellation, with a vision of common future mobile and wireless

technologies. It aims to achieve "alignment of the views on wireless technologies via liaisons with MITF in Japan, the UMTS Forum in Europe, the NGMC Forum in Korea, and the FuTURE project in China". The EU has provided funding within the 6th Framework Program (2002-2006) to projects within the *Wireless World Initiative* (WWI) [242] that builds upon the vision of the WWRF. The WWI was a set of five coordinated research projects[8]. One of these was the *Wireless world INitiative NEw Radio* (WINNER) project [244]. WINNER phase I and II consisted of around 45 partners each, from academia and industry, with well known names such as Alcatel, Chalmers, Ericsson, ETH Zürich, Motorola, Nokia, NTT DoCoMo, Siemens, TU Dresden, to name a few. The WINNER projects were completed some years ago and the final report was published 2008 [96]. To put these projects into relevance of the dissertation, the author was a part of the Wireless IP project (described later), that influenced the WINNER system, which in turn influenced the 3GPP LTE concept, which is what operators are now deploying. With regards to continued research and funding within the area, EU is continuing funding for new projects in the 7th Framework Program and the ICT Future Networks Projects.

Mobile system standardization

Parallel to the research and development efforts is the standardization of mobile systems. The *International Telecommunication Union* (ITU) is "the United Nations specialized agency for information and communication technologies - ICTs" [105]. As such, it is an internationally recognized authority, and its *International Mobile Telecommunications* (IMT-2000) specification sets the requirements for "3G" systems. As mentioned previously, the ITU considers the IMT-Advanced specification to define "4G" systems.

For 3G and 4G, system implementations targeting IMT standards are carried out in the *third generation partnership project* (3GPP). 3GPP is a collaboration of six standard associations and their respective member partners, organized by ARIB in Japan, ATIS in the US,

[8]MobiLife, SPICE, Ambient Networks, WINNER and E2R

CCSA in China, ETSI in Europe, TTA in Korea and TTC in Japan, thus covering a large part of developed nations.

3GPP standardizes and develops the Universal Mobile Telecommunications System (UMTS), a collective term for the 3G technologies developed within the partnership. UMTS is evolved from the GSM specification to reuse components of the core network. The UMTS specifications is published in ongoing numbers of releases. As mentioned in the previous section, the first UMTS release, Release 99, provided packet data services of 384 kbit/s. Later, Release 6 and 7 increased this to up to 14 Mbit/s downstream and up to 5 Mbit/s upstream with the HSPA technology. In UMTS Release 8 and 9, also known as 3GPP Long Term Evolution (3GPP LTE), the CDMA radio interface is replaced with OFDM, as well as other features. This allows for an order of magnitude higher capacity, and prompted operators to market this as "4G" to separate it from the lower capacity 3G offerings, although it is still within the scope of IMT-2000. The most recent UMTS release, Release 10, also known as 3GPP LTE-Advanced, was recently published and approved by ITU as one of two systems (the other being WirelessMAN-Advanced 802.16m) adhering to the IMT-Advanced specification.

Wireless IP

The overall objective of the Wireless IP project [205] was "to develop and to study algorithms and system design for future wireless packet data transmissions beyond 3G" [206]. More specifically, the project investigated the potential of using adaptive transmission on short time scales. The scientific outcome of the project was results in the area of channel prediction, scheduling, link and medium access layer optimization, effects on transport layer protocols (presented in this dissertation), among other aspects. The research results were combined into a 4G[9] system design proposal based on adaptive OFDM. The Wireless IP downlink system, an emulator of the downlink, and

[9]This term was used before standard definitions were available. It is not a claim nor indicates adherence to the IMT-Advanced specification published at a later time.

research results are presented in Chapters 5 and 6. As noted in the Wireless IP final project report [206], ideas, results, and researchers from Wireless IP influenced and worked on the WINNER system concept. The WINNER research in turn formed the 3GPP LTE concept. A quote from the final project report summarizes this influence well:

> Regarding the industrial and societal relevance of these projects, where we made a very significant contribution, we would like to cite Dr. Johan Nyström, head of 4G research at Ericsson, in a comment to collaborators (July 2009): "All new and old Winner people should know that the concept behind LTE originally was conceived and developed in Winner I, started 2004 before LTE even existed. When the necessity of an LTE concept arose in 3GPP, the basic concept was taken from Winner, but downscaled to small BW! A natural platform to develop into an Advanced system of course."

2.4 Internet protocols

There are three main protocols that build the foundation of the Internet; IP [176], TCP [177] and UDP [175]. These belong to the network and the transport layer in the layering model presented earlier. The purpose of these protocols is to enable communication between different hosts and applications running on these hosts. This is achieved by dividing the communicated data into smaller parts (*segments*) of the whole data stream). These segments are prepended with a protocol header that contains control information, explained below. Regarding terminology, the data and headers are sometimes referred to as segments and sometimes as packets. When segments are discussed, it relates to TCP specifics, while packets relate to IP or UDP. A packet is seen as self contained, while a segment is part of a whole.

2.4.1 IP

The main purpose of the Internet Protocol [58, 176] is to provide host addressing. It contains information about the *source address* and *destination address*, as illustrated in Figure 2.13 which shows the IPv4 header. As the number of Internet hosts grow and the IPv4 address space becomes exhausted, the interest to move to the next version of IP, version 6, increases. The main difference between IPv4 and IPv6 is that IPv6 has 128 bits for host addressing, compared to 32 bits in IPv4. The dissertation is throughout focused on the version mainly in use today, IPv4. It is the authors expectation that using IPv4 or IPv6 would make little or no difference to the discussion and research results in this dissertation, apart from the impact of a larger header overhead. Also, in case of TCP-L (presented in Chapter 4), an IPv6 header is presumed to be harder for doing header recovery because of the increased size.

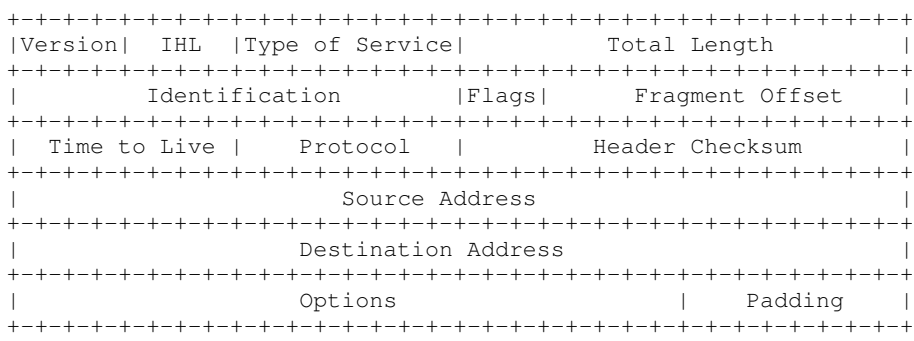


Figure 2.13: The IPv4 header (from RFC 791 [176])

The IP header does not contain information on how to reach the destination[10]. This is handled by the routers that keep tables of destination networks and which routes lead there. Once the packet has arrived at the destination, the receiver can use the source address to send packets back.

IP also provides more information than just addressing. The *type of service* is an indicator of the type of data that is being carried. Via

[10]Actually there is a source routing option, but it is not used mainly because of security issues.

this mechanism, different priorities with regard to delay, throughput, reliability and cost can be defined. This mechanism was not widely used, and these header bits are now used for other quality of service indications in the form of Differentiated Services [33, 167]. IP can perform packet fragmentation, that is dividing a packet into smaller parts, if the packet is larger than the maximum transmission unit size of a network link.

In an interconnected network there may be routing loops. Because of this, IP has a *time to live* counter. It is set by the sender, and each router decrement this counter. When the counter has reached zero, the packet is discarded. This avoids packets getting stuck in infinite loops.

IP was also designed to be extensible. The header contains a variable length option field. New functionality can thus be added by amending the standard, while keeping compatibility with older implementations. In practice, IP options are seldom used due to the design of switching fabric. To attain high router speeds, packets are processed by hardware that inspects the minimal parts necessary to forward the packets to the destined output port. Packets without IP options are handled in this "fast path". With IP options packets enter the "slow path", where the packet is processed by the router CPU which increases delay and reduces throughput.

To protect the header information from accidental modification, the header contains a checksum[11]. An invalid checksum causes the packet to be discarded by a router or receiver. With regard to other aspects of reliability, IP provides none of this. For example throughput capacity, delay, and packet loss are handled on a best effort basis and depend on the traversed routers and links.

2.4.2 UDP

The IP protocol only addresses individual hosts. As a host runs many different applications that may want to communicate, another

[11] The checksum does not provide protection against malicious modification as it is easy for an adversary to recalculate the checksum, compared to a cryptographic signature.

protocol must be used for differentiation. In the IP protocol suite, this is handled by the transport protocols User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP).

UDP [175] is a very simple packet oriented protocol. It only provides multiplexing and checksumming of data. The multiplexing is done via *ports*, where two integers represent the source and destination ports. Applications that wish to receive datagrams then instruct the operating system to deliver datagrams received at a specific port to the application. The UDP header structure is shown in Figure 2.14.

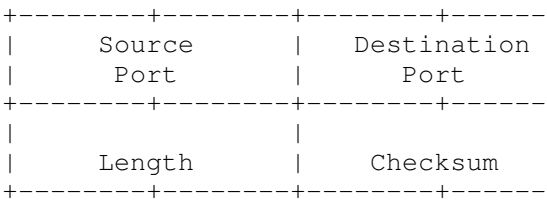


Figure 2.14: The UDP header (from RFC 768 [175])

Like IP, UDP contains a checksum. This checksum covers both the header and data[12], as this header may be the final header before the actual data begins. It is, however, not uncommon to do additional nesting. For example, the Real Time Protocol (RTP) [195], adds an additional header before the data. This header contains for example sequence number and time stamp properties of the data.

UDP, like IP, does not provide any guarantees about delivery throughput, delay or packet loss. This means that any reliability mechanisms must be handled by the application itself. For example, the Domain Name System (DNS) [163] application that utilizes UDP retransmits requests when there is no reply. Another, more problematic scenario is the trap message in Simple Network Monitoring Protocol (SNMPv1) [44]. The trap message is sent in a UDP packet by a monitored device as a response to an event, but the device expects no reply. Thus, if that UDP packet is lost, then the monitor will not know about the event, and the device will not know that

[12]In the case of UDP-Lite [136, 137], the header can have a variable coverage.

the alarm was not received (in SNMPv2 [178] the protocol has been extended with an InformRequest message that provides a notification mechanism like traps, but with confirmations).

2.4.3 TCP

TCP [177] is a connection oriented transport protocol which provides a reliable byte stream to the application layer. Like UDP, it also provides a multiplexing mechanism with ports to allow communication between multiple applications. The structure of the TCP header is shown in Figure 2.15. Besides the port numbers and checksum that

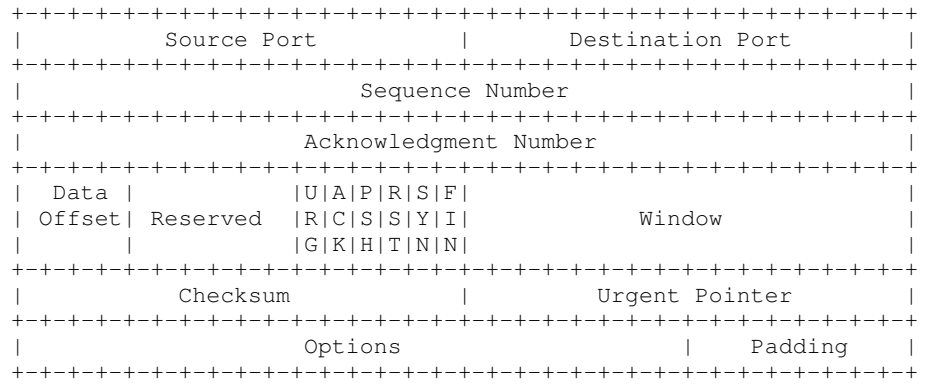


Figure 2.15: The TCP header (from RFC 793 [177])

UDP also contains, the additional header fields are used to facilitate reliable transfer and connection handling.

Application data submitted to TCP is divided into protocol data units, or segments, before transmission. In contrast to UDP, TCP provides reliability by performing retransmissions of lost data. TCP uses an ARQ mechanism based on positive acknowledgments. Each byte is numbered and the number of the first byte in a segment is used as a sequence number in the TCP header. A receiver transmits a cumulative acknowledgment in response to an incoming segment which implies that many segments can be acknowledged at the same time.

To handle connection states, the header contains individual bits to perform setup, closing and reset of the connection. These are marked URG, ACK, PSH, RST, SYN and FIN in Figure 2.15. TCP uses a three-way handshake, where the first packet has the SYN (synchronize) bit set. This instructs the recipient that a new connection is being set up. A reply with the SYN and ACK (acknowledge) bits set is sent. The initiator in turn acknowledges this with a packet with ACK set. After this connection setup, data can be sent in either direction.

To close a connection, either side sets the FIN (finished) bit in a packet. This is acknowledged by the other side. However, as the connection is full duplex, this only indicates that no more data will come from the closed side. The connection is *half closed*. The other side may still transmit more data. When this transmission is finished, the open side sets the FIN bit and the connection is fully closed.

If the connection state gets out of sync, or unexpected data arrives, the connection can be forcefully closed. For example if a TCP connection is active and one host suddenly reboots without closing the connection, packets may still be arriving after boot up. In response, a packet with the RST (reset) flag is sent. This forces a reset or forced closing of the other end of the connection.

To provide reliability, both for the connection handling and data transmission, TCP expects to get acknowledgements of transmitted data. Since IP does not provide any guarantees about packet delivery or even notifications when packets are discarded, this must be handled by TCP. This is done by keeping a *retransmission timer*. The timer is started when a segment is transmitted. If the timer expires before the segment is acknowledged, the segment is assumed to be lost and TCP retransmits the segment. The retransmission timeout value (RTO) is calculated dynamically based on measurements of the round trip time (RTT) [173], i.e., the time it takes from the transmission of a segment until the acknowledgment is received. Since the timer is dynamically adjusted to the estimated round-trip time, it means that it can operate efficiently over networks with both short and long delays. This can be compared to a short static retransmission timer, which would cause frequent unnecessary retransmission

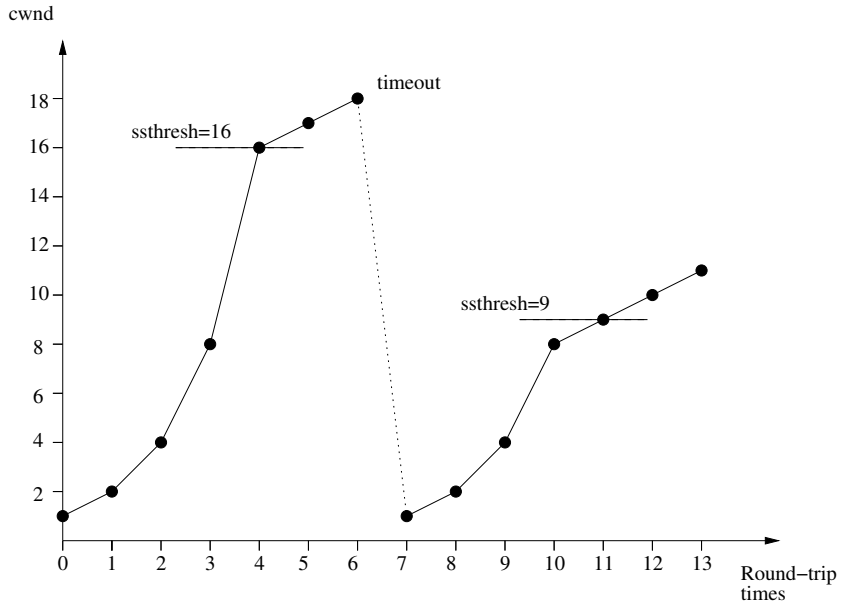


Figure 2.16: The congestion window is increased by one for each received acknowledgement during the slow start phase. When reaching the slow start threshold, the congestion window is increased by one per round trip time. If a timeout occurs, the slow start threshold is set to half of the congestion window, and the congestion window is then set to one segment.

on a long-delay network. Likewise, a long static retransmission timer would cause unnecessary long retransmission delays on a low delay network.

Slow Start and Congestion Avoidance

In 1986 the Internet had its first congestion collapse. The end hosts transmitted more data than the routers were able to handle, and did not lower the transmission rate even though many packets were lost. Hence the congested state persisted in the routers. TCP was therefore extended with mechanisms for congestion control [116] in the form of slow start, congestion avoidance, fast retransmit, and fast recovery [15].

TCP estimates the available capacity in the network by gradually

increasing the number of outstanding segments. TCP is described as a self-clocking protocol, since the sender only transmits segments when acknowledgments are received (the *conservation of packets* principle [116]). The congestion window (`cwnd`) limits the amount of data the TCP sender can inject into the network. The receiver window (`rwnd`) indicates the maximum number of bytes the receiver can accept. The value of the `rwnd` is advertised to the sender, since the receiver includes `rwnd` in the segments going back to the sender. At any moment, the amount of unacknowledged data is limited by the minimum of the `cwnd` and the `rwnd`.

TCP operates in one of two phases, the slow start and the congestion avoidance phases [15,214], depending on the `cwnd` being below or above a slow start threshold (`ssthresh`). Figure 2.16 illustrates this. In the slow start phase (`cwnd` $<$ `ssthresh`), the congestion window is increased by one segment for each acknowledgment received which gives an exponential increase of the congestion window. Slow start is used for newly established connections and after retransmission due to timeout. The congestion window is increased until a timeout occurs or the threshold value `ssthresh` is reached. If a timeout occurs, then `ssthresh` is reduced to half the amount of outstanding data, the congestion window is reduced to one full-sized segment (*multiplicative decrease*), and the slow start phase is entered again. If `cwnd` $\geq$ `ssthresh`, then the slow start phase ends and congestion avoidance is entered instead. During the congestion avoidance phase, the congestion window is increased by one segment per round trip time, which gives a linear increase (*additive increase*) of the congestion window.

The slow start and congestion avoidance algorithms thus tries to find a balance between congestion and performance, by using injected packets as probes and lowering the send rate as packets are determined to be lost. The `ssthresh` is used as a capacity estimate, and when reached, packets are more slowly injected into the network instead of exponentially as with slow start. With less packets injected into the network, the load on the routers decreases, packets can flow through and congestion is therefore avoided.

This handling is termed additive increase/multiplicative decrease (AIMD) and is the TCP standard adjustment algorithm of the con-

gestion window as specified by the IETF. However, in practice there are many other algorithms in use as shown by live measurements presented in [247]. Some of these algorithms are discussed in more detail in Section 2.4.4.

Fast Retransmit and Fast Recovery

The fast retransmit and fast recovery algorithms [15, 214] allow TCP to detect data loss and perform error recovery before the transmission timer expires in some situations. The algorithms increase TCP performance, partly due to the earlier loss detection and retransmission, partly since the transmission rate is not reduced as much as after timeout.

If a segment arrives out of order, the receiver transmits an acknowledgment for the last segment received in sequence. Since this segment already has been acknowledged once before when it was first received (unless delayed acks are used), this subsequent acknowledgment is called a duplicate acknowledgment (dupack). After receiving three dupacks in a row, the sender concludes that unacknowledged data must have been lost. The reason for not reacting after the first dupack is that packet reordering can happen in the network, which would also cause a dupack. Three dupacks is thus a trade-off between reacting quickly to supposed packet loss and not misinterpreting packet reordering as packet loss. After the receipt of the third dupack (the fourth acknowledgment for the same segment) data is then retransmitted directly even if the retransmission timer has not expired. Then `ssthresh` is reduced, as after timeout, to half the amount of outstanding data.

After the retransmission, fast recovery is performed until all lost data are recovered. The congestion window is set to a higher value than after timeout, to three full-sized segments more than `ssthresh`. The additional three segments account for the three segments which triggered the receiver to transmit the dupacks. If more dupacks are received, then the congestion window is increased with one segment for each dupack, since each dupack indicates that one segment has left the network. The fast recovery phase ends when an acknowledgment

which covers new data is received. The congestion window is then set to the same value as `ssthresh`. The effect of this adjustment of the congestion window after fast retransmit and fast recovery is that TCP may enter congestion avoidance instead of first performing a slow start, as is done after timeout.

TCP options

Like the IP protocol, TCP may be enhanced by the use of optional features. Some of the commonly used options which are relevant for TCP in wireless networks are selective acknowledgments (SACK) [156], timestamps [115] and window scaling [115].

The selective acknowledgment option (SACK) [156] improves TCP performance when multiple segments are lost in the same window. With SACK enabled, a receiver can acknowledge up to three non-continuous blocks of received bytes in the same acknowledgment. The sender then knows which segments are missing and can retransmit only those.

The timestamps option [115] provides an additional means to identify segments and their acknowledgments. A twelve byte timestamp is added to outgoing segments and the receiver adds the same timestamp to the acknowledgments going back to the sender. If the timestamps option is enabled, then the sender can sample the round trip time with a higher frequency, which gives a more accurate round trip time estimation. This is especially useful when large windows are used, since the round trip time can be estimated more often than only once per window.

The window scale option [115] can be used in order to utilize the network capacity between the sender and the receiver more efficiently. The bandwidth-delay product may be larger than the maximum value of the header field for the advertised receiver window (with 16 bits, $2^{16} = 64$ kbytes). This means that the transmission is limited by the advertised receiver window, although the network can transport more data. With the window scale option, a larger window can be used, since it is possible to advertise a receiver window of 32 bits.

Other mechanisms

Besides using options in the header, there are other aspects of the protocol behavior, such as limited transmit, increasing the initial congestion window and adjusting the maximum transmission unit.

Limited transmit [12] is a modification to the loss recovery algorithm in TCP. Without limited transmit, TCP segments are only transmitted when there is room in the congestion window. The limited transmit mechanism allows TCP to transmit a new segment when the first dupack arrives, even if the congestion window normally limits such a transmission. The rationale for this is that the arrival of a dupack indicates that one segment has reached the receiver and left the network. By transmitting a new segment, TCP checks if the network is congested or not. If the new segment reaches the receiver, the probability of fast retransmit is increased, since the receiver transmits a dupack in response to the new segment.

Limited transmit improves TCP performance when the transmission window is too small for fast retransmit and fast recovery to be triggered. For example, if the window is only two packets, the sender may only transmit two packets and therefore three dupacks can not be generated. Limited transmit allows for injection of more packets which may lead to more dupacks, which then trigger fast retransmit.

The setting of the initial congestion window affects the TCP performance, and has been increased from 1-2 segments to 2-4 segments (depending on segment size, corresponding to 4380 bytes). See [13] for a discussion and [15] for the latest standard. An increased initial congestion window is beneficial in networks with long round trip times, and for short flows where data could be delivered using fewer round trip times leading to a shorter total transmission time. There are arguments to increase this further to 10 segments [62], to match the increased average length of short flows since the last window increase [15]. Instead of a fixed initial congestion window, there is also an IETF draft suggesting a dynamic window size [228].

The maximum transmission unit (MTU) is decided by the sender, and depends on the sender connection type. For a wired connection, this is typically 1500 bytes in Fast Ethernet. Larger MTUs

can be used, for example WiFi accepts up to 2304 byte data units (MSDU) [102]. However, unless the whole path supports the actual transmission unit, IP fragmentation is needed, which causes an overhead. Individual fragments may also be lost, depending on the link reliability properties, causing the whole packet to be discarded. On links with a high error rate, it is better to use a small segment size that increases the probability of successful transmission.

2.4.4 TCP variants and naming

There are many implementations of TCP, some of which are considered as baseline TCP implementations. Three TCP implementations in the BSD operating system, named Tahoe, Reno and NewReno, are the ones most commonly referred to in the literature. TCP Tahoe is the original 4.4BSD implementation, including the congestion control scheme devised by Jacobson [116], i.e., the slow start and congestion avoidance algorithms mentioned earlier. The addition of the fast retransmit and fast recovery algorithms [16] to TCP Tahoe is called TCP Reno. TCP NewReno [73] improves upon TCP Reno by changing thresholds in the fast recovery algorithm and avoiding a scenario where multiple retransmits can occur after timeout.

However, as TCP additions and modifications are continuously proposed and implemented, TCP implementations are not that easily classified to these three original implementations. For example, Linux (2.6.19 and forward) by default uses the *CUBIC* TCP congestion control, which does not rely on acknowledgements to increase the congestion window as the original TCPs. Instead the delay since the last congestion event is used to grow the window size with a cubic function [95]. Microsoft Windows Vista and forward includes *Compound TCP* (CTCP) [222] congestion control. CTCP extends TCP Reno, and uses a delay-based congestion window in addition to the loss-based congestion window. The sending rate is increased faster when there is low delay, implying a under-utilized network. When network queues increase, the delay increases and the sending rate is reduced to avoid congestion.

Linux TCP implementation

The Linux TCP implementation is built on a modular design. The core TCP specification is followed to allow interoperability, but for example the congestion control algorithm can be dynamically changed, along with other parameters that can be adjusted while still maintaining the TCP semantics with byte ordering and reliable transport. To give an example, the following algorithms are implemented in the Linux 3.0 kernel, and easily activated by loading the respective kernel module: `bic` (Binary Increase Congestion control) [245], `cubic` (Cubic Congestion control; as discussed in the previous section, this is the default congestion control algorithm in Linux 2.6.19 and forward) [95], `vegas` (TCP Vegas) [36], `highspeed` (Highspeed TCP) [71], `htcp` (H-TCP) [139], `hybla` (TCP Hybla) [41], `illinois` (TCP Illinois) [146], `lp` (Low Priority) [133], `scalable` (Scalable TCP) [122], `veno` (TCP Veno) [53], `westwood` (TCP Westwood+) [92] and `yeah` (Yet another high speed TCP) [21].

Most of these algorithms are for high speed networks and large bandwidth delay products, but there are some algorithms aimed at wireless networks, such as Hybla, Veno and Westwood+ (Westwood+ is discussed in the next chapter on TCP end-to-end optimizations). Other than providing different congestion control algorithms, Linux also performs other optimizations. For example, a retentive caching of TCP control information is kept for destinations recently communicated with. This allows the sender to more quickly determine the available bandwidth, by assuming it has not changed much since the previous connection. Further, the sender and receiver buffer sizes are automatically adjusted. Normally buffers of 32 to 64 kbytes are used, which can limit the performance when the bandwidth delay product is large. Linux allows setting a minimum, a default, and a maximum allowed buffer size. The actual size is then automatically adjusted within the specified limits for each connection. Relating to the earlier discussion on increasing the initial congestion window [62], the Linux kernel changed this setting from 4 to 10 in early 2011, and this setting is used by kernels since version 2.6.39.

2.4.5 Other protocols

Although TCP and UDP are the dominant transport protocols, there are other transport protocols in use. One example is the *Stream Control Transmission Protocol* (SCTP) [216]. Compared to TCP, SCTP is message based (like UDP) instead of byte based, and can utilize multiple streams within one association. This means messages can be sent on different streams and arrive as soon as possible, avoiding the head-of-line blocking that can occur in a single-stream transport protocol such as TCP. An interesting feature of SCTP is multi-homing. An SCTP association can be created between multiple end points, with different IP addresses. This can be useful in a wireless network, where roaming and handover can cause the IP addresses to change. With TCP both end points must keep their addresses for the duration of the connection, while in SCTP multiple addresses can be used.

Another example is the *Datagram Congestion Control Protocol* (DCCP) [125]. Applications that traditionally use UDP, for example in media streaming, have to implement their own "connection" setup, and congestion control (if this is at all considered). DCCP provides many of these functions for unreliable datagram transmission, i.e., connection setup, handling ECN congestion notifications, and congestion control mechanisms that use a fair share of the available capacity when DCCP/TCP flows are mixed [72, 75].

2.5 TCP in wireless networks

TCP has been found to perform poorly in some wireless networks, reported by [25, 47, 65] and others. The poor performance is mainly due to the fact that TCP cannot distinguish problems that typically occur in wireless networks from congestion. The congestion control algorithms in TCP are based on the assumptions that data is lost mainly due to congestion and that data loss due to transmission errors is rare [116]. Therefore, data loss is interpreted as a signal of congestion in the network. Even in a wireless network, where data loss may not be related to congestion, data loss still signals congestion to the sender.

TCP segments may be lost if the radio conditions are poor and the link layer protocol provides a low reliability. After some retransmission attempts the link layer protocol gives up and leaves further error recovery to TCP. Handover events may also lead to data loss. A whole window of data may be lost due to handover. Data loss due to an low reliability link layer or a handover, may cause a timeout event followed by slow start or three dupacks followed by fast retransmit and fast recovery. In either case, the congestion control action taken by TCP is unnecessary. Directly after a loss event due to a bad radio condition, the radio quality may become good again and support the previous transmission rate.

TCP may also misinterpret a sudden increase in the round trip time (a "delay spike") as data loss [140]. If the delay is long enough for the retransmission timer to expire before an acknowledgment is received, then TCP misinterprets the delay as an indication of data loss due to congestion. The delayed data is unnecessarily retransmitted and TCP enters slow start. This phenomena is known as a spurious timeout [94]. A highly variable round trip time can also lead to a large RTO, since the RTO is based both on estimates of the round trip time and on variations in the round trip time. If the RTO is large, then TCP reacts slowly to data loss. Variations in the round trip time can be caused by link level retransmissions of a wireless link. If the link layer frames that contain parts of a TCP segment must be retransmitted because of a poor radio environment, then the whole segment is delayed. Round trip time variations may also be caused by handover or competing traffic. Queuing in routers, base stations, and other intermediate nodes may also lead to a long round trip time. A long round trip time may cause low throughput and under-utilization of the network, since it takes a number of round trip times before the congestion window reaches the capacity of the network. TCP performance is degraded, especially for short lived flows, which transmits a small amount of data.

TCP can also experience unnecessary congestion control because of packet reordering. Although no packet is lost in this case, it is still interpreted as a packet loss event. This scenario can occur if the wireless link does not provide an ordered service, which is a reason

why many link layers provide an ordered service. On the other hand, buffering packets to provide an ordered service causes increased delay variations and complexity in the link layer. Issues of reordering and TCP robustness modifications to these is further discussed in [29, 51, 101, 135]. Another reason for reordering is multipath routing, where packets may be sent on different paths with varying delays.

As mentioned earlier, TCP is a self-clocking protocol that acts upon the receipt of acknowledgements. With the varying channel condition of a wireless link, this can cause compression of acknowledgements on the uplink. These then arrive in bursts, causing corresponding burstiness of outgoing data. Packets in this burst can then cause congestion or experience delay spikes on the downlink, leading to the problems discussed above.

Finally, the capacity of the wireless link can fluctuate as the strength of the radio signals vary. This leads to varying delay, which inflate the RTO timer or can cause spurious timeouts. The inherent rate variations can also cause non-optimal channel utilization or cause buffer overflows [49, 94, 190, 234].

Chapter 3

Related work

On most network paths, loss due to damage is rare ($\ll 1\%$) so it is probable that a packet loss is due to congestion in the network.

— Van Jacobson, 1988 [116]

The previous chapter presented the background of wireless communication, the major Internet protocols, and discussed the problems that arise for TCP in wireless networks. There are many proposals that aim to improve the TCP performance, at the end hosts or in the communication path. This chapter presents an overview of such optimization proposals. Further, a number of performance evaluations of TCP in deployed cellular 3G networks is presented. The chapter continues with an overview of network emulation, and concludes with a summary.

3.1 Proposed optimizations[1]

There are many proposed TCP optimizations for wireless networks in the literature, and to present these in a structured manner they

[1]Parts of this section were published earlier in [5, 240]. This is a revised version.

are here categorized into four groups: *end-to-end*, *link layer*, *explicit notification* and *split connection*. This is illustrated in Figure 3.1. The categorization is done from the key components involved in the communication (sender, base station, receiver), and is a permutation of the paths between them. The actual communication path does include many more parts, but these form the considered system abstraction level.

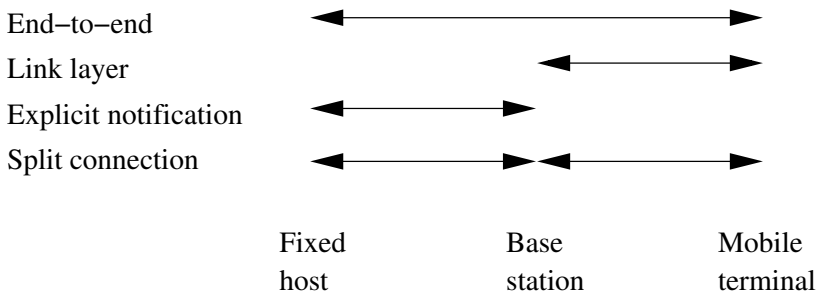


Figure 3.1: Division of optimizations

3.1.1 End-to-end

Optimizations can be placed entirely in the end-points to avoid adding complexity to the network. This benefits from the *end-to-end arguments* [191] (part of the architectural principles of the Internet [43]). The proposals in this section have been further grouped as loss differentiating, rate-based and others.

Loss differentiation

As mentioned, non-congestion related packet loss invokes inefficient congestion control. Therefore a number of performance improving proposals aim to differentiate between non-congestion related wireless link loss and congestion related loss. Coupled to the differentiation is notification and response, i.e., how the TCP sender is notified and how the notification should be responded to. Proposals here operate on the end-to-end path, while Section 3.1.3 discusses explicit notifications originating within the network.

Wireless errors that result in residual bit errors in packets are discarded by TCP because the packet checksum is incorrect. TCP-HACK [26] adds two options to TCP; the header checksum option and the header checksum ACK option. If the options are enabled, an extra checksum of the TCP header is added, allowing the receiver to distinguish between errors in the header and the payload. The sender is informed of a received but broken segment via the checksum ACK option, and can retransmit the segment without initiating congestion control. Both sender and receiver must be aware of the modification. Experiments with a HACK implementation in Linux have been performed and compared to NewReno and TCP SACK. These show that HACK is able to keep packets flowing to a larger degree than using SACK and to a larger degree than using NewReno alone. The best performance is achieved by combining both HACK and SACK.

The packet corruption scheme CNO/CDO [239] is a similar proposal. Here, a checksum of parts of the TCP header is added as a TCP option. When a TCP checksum fails due to packet corruption, the extra checksum is verified to determine if the header is intact. If it is, the header content is trusted and a corruption notification option is used in the returning acknowledgement segment.

LN-MAC [184] is similar to CNO, but the detection is performed in the medium access layer instead of a separate header checksum as in CDO. Simulations of a 802.11 WLAN with CNO, LN-MAC and non-optimized TCP show large improvements for both optimizations. An interesting effect is shown when link layer retransmissions are enabled. With CNO, multiple acknowledgements for the same packet may be sent, even though it is only locally retransmitted on the wireless link. This causes the sender to detect dupacks, leading to worse throughput than non-modified TCP.

A similar approach with an extra header checksum as a TCP option is used by an corruption-aware adaptive increase and adaptive decrease algorithm [54]. The checksum-based loss differentiation scheme [83] uses existing checksums already available in the protocol stack to detect corrupt packets. TCP-L [6] described in the next chapter delivers corrupt data to the application level and acknowl-

edges packet reception.

Another approach to differentiate losses is to use RTT, delay or inter-arrival time measurements. For example TCP-CERL [64] uses RTT measurements to estimate queue lengths in the bottleneck router, and uses this to infer a congestion level. When a packet loss occurs and the estimated congestion level is above a threshold, the loss is classified as a congestion loss and congestion control is initiated. If the congestion level is below the threshold, the loss is classified as a non-congestion loss and the packet is retransmitted, without reducing `ssthresh` or `cwnd`. This is similar to the rate-based schemes discussed below, but with the conceptual distinction that TCP-CERL still uses the traditional ack-clocking mechanism to determine the transmission rate, although the ack rate is used to differentiate the cause of packet losses.

Further loss-differentiating proposals are discussed in [31, 32, 80, 127, 131, 142].

Rate-based schemes

Standard TCP uses incoming acknowledgements to adjust the sending rate according to the slow start or the congestion avoidance algorithm described earlier. An alternative approach is used by TCP Westwood+ [92], where the path capacity is determined from the arrival *rate* of acknowledgements, and this determines the sending rate. This provides a faster error recovery compared to standard TCP because the rate estimation is not multiplicatively decreased but smoother adjusted based on the acknowledgement arrival rate and inferred available capacity. This is beneficial for wireless environments with non-congestion related packet loss; whereas standard TCP reacts drastically to even small amounts of packet loss, TCP Westwood+ keeps up the sending rate. The authors note that simulations of Westwood+ show improvements of the goodput compared to NewReno TCP when bursty losses affect the wireless channel, and that fewer link layer retransmissions are required by Westwood+ to achieve full utilization [92].

Another example is the Wireless Transmission Control Protocol

(WTCP) [201], aimed at improving the performance in cellular packet data networks. As TCP Westwood+, WTCP uses a rate based approach to control the transmission rate. Inter-packet separation at the sender and the receiver is used as the primary metric for transmission rate calculations. Like TCP Westwood+, this means that WTCP is more resilient to non-congestion related packet loss, thereby improving performance in wireless networks.

TCP Real [250] is a rate based scheme in which the receiver controls the transmission rate at the sender. The receiver uses changes in the rate of incoming segments to compute the congestion window the sender should use. If the rate of incoming segments is decreasing, then this is taken as an indication of an increasing load in the network and therefore the `cwnd` should be decreased accordingly. After data loss, the `cwnd` is adjusted to the network conditions sooner than in standard TCP, since the receiver includes estimates of the `cwnd` in the acknowledgments that go back to the sender.

Others

There are many end-to-end TCP optimization proposals that are not easily grouped, and a selection of these is presented here.

Freeze-TCP [89] is a mechanism to improve the performance of TCP in wireless environments in which handover frequently occurs. By exploiting the properties of the advertised receiver window, a TCP connection can be frozen. If the receiver sets the receiver window to zero, then the sender leaves its `cwnd` unchanged until the receiver advertises a new receiver window. Just before handover occurs, the mobile station freezes the TCP connection by advertising a receiver window of zero. This prevents segments from getting lost and unnecessary congestion control action to be taken by the sender. When the handover is completed, the receiver sends an acknowledgment which opens up the window again. The transmission then continues at the same rate as before. A development of Freeze-TCP is ATCP [199], where the mobile terminal is also considered to be transmitting. Besides sending zero-window advertisements, a disconnect event causes ATCP to save the `cwnd` and `ssthresh` in the terminal, and restore

these when connectivity is restored.

A delayed dupack scheme proposed in [231] is an end-to-end scheme that imitates Snoop (described later). The third and subsequent dupacks are delayed while the base station performs link level retransmissions. This prevents the sender (fixed host) from taking congestion control action while the base station retransmits data over the wireless link. The third and subsequent dupacks are delayed for a predetermined time interval. During this time the receiver may receive the missing data and transmit a cumulative acknowledgment instead. The delayed dupacks are then discarded. If the missing segment is not received, then the delayed dupacks are instead released when the timer expires.

TCP-DCR [30] is also delaying the effect of dupacks, but on the sender side. When three dupacks have been received, TCP-DCR waits for a time period τ before initiating congestion control. The idea is to allow more time for link level retransmissions of the lost packet before deciding that the loss was due to congestion or wireless conditions. If no acknowledgements have been received during time period τ, congestion control is initiated as usual. A similar approach is taken by TCP-NCL [134] where two timers are used. After the first timer expires, the packet is retransmitted, but no change is made to `cwnd` and `ssthresh`. After a second timer expires, the packet is again retransmitted and congestion control initiated.

The Eifel algorithm described in [151, 152] allows the sender to detect whether an already initiated error recovery action is necessary or not. When the first acknowledgment that covers previously unacknowledged data arrives, the sender can determine if this is an acknowledgment of the original segment or of a retransmission. The algorithm uses the timestamps option to match acknowledgments with segments. If an acknowledgment of a segment is for the first transmitted segment, then the retransmission is spurious and there is no reason for the sender to reduce the transmission rate. The first segment was not lost due to congestion, but was delayed before it arrived at the receiver. Eifel therefore restores the `cwnd` and `ssthresh` to undo the congestion response.

The last three optimizations are usable when reordering can oc-

cur. For example when an unordered link layer is employed or when packets are routed via multiple paths.

Conclusion

The end-to-end proposals are based on various ideas. Loss differentiation proposals try to separate non-congestion related wireless packet loss from congestion loss, and respond appropriately. The rate based proposals try both to avoid congestion and to recover quickly from non-congestion losses over the wireless network.

An interesting observation is that some of these differentiation and rate-setting techniques use the arrival rate of acknowledgements. For example, using an optimization that regulates acks at the link layer could cause unintended effects with an end-to-end optimization that uses ack timing measurements.

3.1.2 Link layer

The purpose of link layer optimizations is to hide the properties of wireless communication to make it similar to wired communication. The techniques for this are further grouped into frame retransmissions, local TCP retransmissions, and regulating TCP control information.

Frame retransmissions

A common method to provide a reliable delivery over an unreliable channel is to perform link retransmissions (ARQ). Typically the delay of a wireless link is much shorter than the total end-to-end delay of the whole communication path, which implies that it is more efficient to do local link retransmissions.

RFC 3366 [68] provides advice to link designers on how retransmission strategies should be designed when the link carries Internet traffic. Of interest is the discussion of retransmission persistency, i.e., should few, many or an indefinite amount of retransmission attempts be performed. Low persistency reduces the risk of interfering with the retransmission timer, while a high persistency introduces delay

and delay variations. It is also noted that different traffic types benefit from different levels of persistency. For a bulk transfer it is more important to have reliable delivery, and less important to have short delay or small delay variations, as long as there is data buffered.

Rather than having a static link retransmission limit, [153] proposes a dynamic limit set by a TCP aware algorithm. Depending on the TCP retransmission timer, segments are given different priority to avoid timeout which would cause the TCP sender to enter slow start. It is, however, unclear how the TCP retransmission timer would be available to the algorithm, since these reside in different parts of the network. It also assumes that a standard TCP is used (experiments used TCP Reno), whereas there might be side effects when another optimization is used that calculates the timeout value differently, for example the rate-based approaches described earlier.

A concern about high retransmission persistency is the introduction of delay and delay variability. This is tightly coupled to the retransmission delay. More retransmissions can be done with a shorter retransmission delay compared to a longer retransmission delay, to reach the same delay budget. As noted by [138], "TCP and RLC[2] retransmission mechanisms hardly ever compete", and the maximum amount of retransmissions should be used. The argument of using persistent link retransmissions is also supported by [230], as well as the experiment results in Chapter 6, while [200] finds that the optimum value in their investigated UMTS network is around 10.

Retransmissions are also commonly combined with error correcting codes, such that the probability of decoding received packets increases with each retransmission. Hybrid ARQ [150] adds error correcting codes to each transmission, and may also use chase combining (see Chapter 6) to combine retransmissions and reduce the noise component. Hybrid Type II ARQ [150] uses incremental redundancy where each retransmission adds more and more error correcting codes, which can be used with previous transmissions. These techniques reduce the amount of needed retransmissions. It is, however, at the expense of additional transmission, computational and

[2]Reliable Link Control is the retransmission mechanism used in GSM/GPRS.

complexity overhead.

Local TCP retransmissions

Apart from performing link frame retransmissions, entire packet data units, i.e., TCP packets, can be retransmitted. This requires the base station to be TCP aware, and to also buffer packets. A frequently referenced early technique is *snoop* [25]. An agent in the base station "snoops" on the passing TCP traffic and caches packets destined for the mobile terminal. If duplicate acknowledgements are sent from the mobile terminal for segments that have passed the base station, snoop assumes that they are lost on the wireless link and performs a local retransmission. The duplicate acknowledgements from the terminal are discarded to avoid triggering congestion control in the sender. In [25], measurements of snoop in a test bed showed that for bit error rates worse than $5 * 10^{-7}$, snoop gives a improvement factor of 1 to 20 (depending on bit error rate) over non-modified TCP. This was done over a 802.11 WLAN without (traditional) link layer retransmissions. Later simulations with snoop over GPRS have, however, shown that TCP performance is not enhanced since the link retransmission delay is on the same magnitude, or larger, compared to the delay in the wired network part [187].

WTCP [183] improves upon Snoop with regard to the delay introduced by performing retransmissions. The added delay interferes with the TCP round-trip time estimation. WTCP mitigates this by adjusting TCP timestamps on the local retransmissions to compensate for the added delay.

Local TCP retransmissions are also prevalent in split connection type optimizations, where the TCP connection is terminated in the base station. Split connection optimizations are discussed in a later subsection.

Regulating control information

Other common techniques involve affecting passing TCP packets to modify the control information that they convey, explicitly or implicitly. With acknowledgement rate control in 3G networks [4], the TCP

acknowledgements on the uplink are adjusted relative to the downlink buffer utilization. This way, a TCP sender that uses ack clocking to decide the sending rate (i.e., most standard TCP's) will slow down if the returning acknowledgements are artificially delayed. A similar approach of regulating acknowledgements is found earlier in [47]. Here, acknowledgements are released only when there is buffer space available for downlink transmission.

Window regulation [48,49] is another technique that avoids buffer overflow in the base station. If the sender is operating in receiver-window limiting mode (i.e., `cwnd` > `rwnd`), the base station modifies the receiver window of returning acknowledgements, according to the available downlink buffers. In simulations, the author found that window regulation and ack buffering to absorb channel variations improved TCP SACK performance by 100%.

By introducing random delays in the communication path [124], these delays contribute to an increased delay variance in the round-trip time calculation, and the TCP sender increases its timeout threshold accordingly. This can "significantly decrease the number of timeouts", and leads to an increase in throughput, according to numerical results. By avoiding the spurious timeouts, the authors note an 8% throughput increase for their considered network scenario. Possible drawbacks with a high RTO, such as longer time to react to true timeout events, are not discussed.

TCP-Acknowledge [233] reduces the round-trip time over wireless links which then leads to increased TCP throughput. The method is also claimed to make the link more robust and to provide more bandwidth for uplink transmission. The base station monitors the wireless transmissions. When a TCP packet is successfully delivered, the base station itself sends an acknowledgement. The mobile terminal abstains from sending acknowledgements since it is aware of this scheme. The acknowledgement transmission is then faster than if the mobile station would send the acknowledgement, leading to the round trip time reduction claimed by the authors. It also eliminates the risk of losing the acknowledgement on the wireless uplink. Since no (pure) acknowledgement packets are sent on the uplink, more bandwidth becomes available for data transmission. It is not

discussed how receiver-specific fields are handled. For example the receiver window is based on the available amount of buffer space, which is unknown to the base station.

Conclusion

The link layer optimizations operate on the link or in the base station to improve the performance of TCP. Some use local link level retransmissions to minimize packet loss due to the wireless part of a connection, while some inspect and modify the traffic that passes through the base station. All the proposals preserve the end-to-end semantics of TCP, although it could be argued that modifying the receiver window (as is done with the window regulator [48, 49]) is changing the semantics of the field. It no longer indicates the available buffer space at the receiver, but is a link quality or buffer indicator of the network. For the proposals to work, TCP data and acknowledgments are required to pass through the same base station (or other intermediate node in which the optimizations are implemented). In approaches that perform local TCP retransmissions, Snoop and WTCP, the base station is required to process information in the TCP header, which is based on the assumption that each TCP segment is encapsulated in a link layer frame. This is usually the case in WLANs, but other wireless networks operate differently. If the link layer frame used over the radio interface is to small to encapsulate a TCP segment, as in many WWANs, then the link layer proposals could operate on the logical link control layer instead (provided that a TCP segment fits into a frame). Also, if TCP headers are encrypted due to security on the IP layer (IPsec) [123], a TCP-aware approach will not work.

It is interesting to note that some of the approaches may become problematic when combined with each other. For example, with persistent link ARQ, techniques such as Snoop will have no effect as packets will not be lost on the wireless link. Schemes that try to differentiate wired and wireless losses (discussed earlier) will not be useful. Artificially changing the ack rate can also cause unintended effects in combination with other approaches that modify TCP in the end points, such as rate-based schemes.

3.1.3 Explicit notification

As TCP interprets all losses as congestion in the network, various loss differentiation and notification schemes have been proposed to enable the TCP sender to determine if the cause of a data loss is due to congestion or not. Discussed so far, these have used end-to-end signaling. An alternative is to put more intelligence in the network; explicit notifications from the network (i.e., base stations, routers) can then be used by the end-nodes to employ a more informed congestion control.

The standard mechanism for early (i.e., before losses occur) congestion notification in wired networks is *explicit congestion notification* (ECN) [182]. ECN uses bits in the IP header which routers set to indicate when congestion is experienced (CE). The receiver detects that congestion is experienced and sets an echo flag (ECN-Echo) in the acknowledgements to the sender. The sender in turn reduces the sending rate like the packet had been dropped (but does not retransmit the packet), and sets a flag (CWR) to indicate that the congestion window is reduced.

A similar approach for WCDMA wireless networks is the *explicit rate change notification* (ERCN) [234]. Here, the varying rate over the link is monitored by the base station. If the transmission rate drops, ECN is used to notify the sender of "congestion", and the sending rate would therefore be reduced to better match the decreased link transmission rate.

Explicit Loss Notification (ELN) [24] can be used if the mobile host is the sender. The base station examines TCP headers and keeps track of the sequence numbers. A hole in the sequence space indicates that data must have been lost over the wireless link on its way from the mobile station to the base station. The fixed host (the receiver) transmits a dupack in response. The base station sets an ELN flag in the dupack header before the dupack is forwarded to the mobile station. The mobile station then retransmits the lost segment but does not reduce its congestion window.

Syndrome [52] is a modification of the base station to enable detection of packet loss over the wireless link. The base station counts

the number of packets it relays, and includes this counter as a TCP option. This counter is called a "syndrome", and is used together with the sequence number to determine if a loss occurred in the wired or wireless part of the network. If there is a gap in the syndrome counter, this indicates that packets were lost on the wireless part. Gaps in sequence numbers, but not in the syndrome, indicate that segments must have been lost in the fixed network. Explicit loss notification [24] is then used by the receiver to inform the sender about the loss. For more examples of explicit notification solutions, see [23, 88, 130, 141].

Conclusion

Compared to the end-to-end and link layer approaches, the explicit notification proposals expose information to the communicating endpoints regarding various conditions and causes of loss. This does not solve the problem with the higher unreliability of the wireless network, but since the sender is aware of the conditions it can avoid triggering congestion control and therefore improve the performance.

3.1.4 Split connection

The idea of split connection approaches is to divide each TCP connection into two separate connections at an intermediate node. By splitting connections, congestion related losses in the wired network are unaffected by wireless losses. Commonly, split connection proposals use TCP, or modified versions of TCP, also over the wireless link. Another protocol stack may also be used over the wireless link, for example as is done in WAP [237].

Indirect TCP (I-TCP) [22] is an example of an approach where a separate TCP connection is established between the fixed host and a *mobile support router* (MSR), and another TCP connection between the MSR and the host in the fixed network. This way, communication problems over the wireless link is handled separately, and errors can be recovered faster due to shorter round trip times and a faster control loop. I-TCP also has provisions for mobility. When a mobile terminal

changes cells, the TCP state is transferred to the mobile support router of the new cell. This enables a seamless handover.

In [46] a mobile proxy solution, TCP `cwnd` clamping, is proposed to improve TCP performance over GPRS. The proxy avoids slow start for the connection over the wireless link. In order to utilize the full capacity of the wireless link, the `cwnd` is set to an estimate of the bandwidth-delay product. As a simple means to limit the buffer requirements in the proxy, the advertised receiver window is modified in the acknowledgments transmitted over the wired link to the fixed host.

Another common type of split connection is application level proxying [34]. The proxy is processing all data on the application layer, and the data can therefore be adapted for the mobile device. For example, images can be down-sampled to suite a smaller screen, or uncompressed data can be compressed. Such a proxy can also cache requests, which further improves performance if the proxy uplink capacity is limited and the same object is requested multiple times, perhaps by multiple users.

A recent investigation of the use of middleboxes in cellular networks [236] finds that more and more carriers are deploying proxy solutions and network address translators. The paper shows the effect of these network policies on performance, energy and security.

Further examples of split connection approaches are available in [56, 120, 161, 186, 246, 248].

Conclusion

By splitting a TCP connection, it is possible to discriminate data loss over the wireless link from data loss that occurs over the fixed network. The proposals have similar disadvantages as the link layer proposals presented above. All data to and from the mobile station is required to pass the same base station (or some other intermediate node which splits the connection). The end-to-end semantics of TCP are not preserved, and the base station must have a buffer for intermediate storage of received but not yet transmitted data, which may never be delivered if the terminal has gone out of reach. It may

also happen that data is acknowledged to the fixed host, but the intermediate node crashes before delivery to the mobile terminal.

3.2 Performance evaluations

The presented optimizations in the previous section often included performance evaluations along with the description of the optimization to illustrate the attainable performance gain. The majority of these optimizations are evaluated by network simulation, and most compare the optimization with a standard TCP, i.e., not compared to other optimizations.

Of interest is the performance of the wireless networks itself. This provides an understanding of how standard (and sometimes optimized) TCP performs in existing systems. There are a number of investigations of GSM/GPRS, UMTS/3G, LTE, and EV-DO networks, using analytical modeling or simulations. For a selection of these, see [20,38,45,57,97,99,100,143,154,189,192,218,232]. We next consider evaluations and measurements of deployed systems in more detail.

Measurements of a live 1xEV-DO networks is published in [147]. The authors note that previous reports assumes channel errors as the main cause of transport layer performance degradation, while they have verified that these are rare in 3G networks due to turbo codes and rate adaptation. This is noted to instead lead to high delay and rate variation as the main cause of degradation. To see these effects, channel measurements are taken to characterize the stationary and mobile channel quality and variations. Stationary and mobile terminals are found to have comparable bandwidths, while mobile terminals (naturally) experience a greater variability. Interestingly, measurements in co-located stationary terminals are found to experience different independent variability of the channel conditions. Beside channel measurements, the throughput of a number of TCP variants is measured. TCP Reno, Cubic and Westwood are noted to be reasonably efficient given protocol overhead, reaching $85\% - 89\%$ utilization of the channel capacity and TCP Vegas reaching 75% uti-

lization. Factors behind this level of utilization are low frame error rates, below 0.01%, and sufficient transport protocol buffering. The buffering by Vegas was smaller than the others, leading to idle periods, which explains the lower channel utilization.

The measurements are also interesting from an optimization comparison perspective. With concurrent traffic, it is shown that Reno and Westwood reach about a quarter each of the available bandwidth, but that Cubic gets $> 30\%$ and Vegas $< 20\%$ of the share. This indicates an unfairness issue when different protocols are involved. This is further related to concurrent real-time UDP traffic and the delay caused. Delay is shown to be proportional to the unfairness displayed earlier. The UDP traffic concurrent with Vegas has the lowest delay, and the UDP traffic concurrent with Cubic has the highest delay. To relate this to actual numbers, Vegas reaches 250 kbyte/s with a 22 ms UDP delay, and 310 kbyte/s with Cubic with a 78 ms UDP delay (numbers are approximate).

In [121], TCP and UDP over WCDMA and HSDPA are evaluated. They find that the maximum goodput matches that of the advertised throughput of the respective network (384 kbit/s for WCDMA and 1 Mbit/s for HSDPA (operator limited)), and also that the newer HSDPA has much lower delay and higher capacity than WCDMA. Regarding measurements in live networks, the paper makes an interesting observation regarding the experiment controlability: "measurements where the data rate dropped suddenly were ruled out from the analysis, as they were understood to be another HSDPA user entering the cell". This points to the difficulty of using live networks for measurements. The existence of other users in the network was inferred from sudden drops in data rate; perhaps the drop had other causes than the assumed? What if another user only used part of the available bandwidth; where would the limit be for discarding or accepting a measurement?

Paper [180] compares TCP and UDP over live HSPA, HSDPA-only and WCDMA networks. Like the study above, they achieve TCP goodput that matches the advertised throughput, except for HSDPA which achieves 2.8 Mbit/s instead of the maximum 3.6 Mbit/s. The authors speculate that other concurrent users might be the cause.

The paper also presents delay figures, where the WCDMA downlink has a mean delay of 98 ms ($\sigma = 6.8$ ms), which in HSDPA is reduced by more than half to 40 ms ($\sigma = 4.9$ ms).

To summarize, traffic in modern deployed networks are reported to function quite well, based on measurements in live networks. This is also in line with the experimental evaluations in this dissertation (cf. Chapter 6), and is attributed to the higher data rates, lower error rates and lower delay of HSPA and EV-DO compared to previous systems such as WCDMA and CDMA2000.

3.3 Network emulation[3]

As mentioned in the introduction chapter, there are a number of alternatives when performing network research and evaluations. These range from purely analytical modeling and calculation, to measurement on live systems as reported in the previous section.

Measuring in live networks is an attractive route, since that captures what a user would experience and includes all aspects of the system, some of which may be lost if simulation is used. However, it requires that there is a system available; some may only be in the design stage, with no real system implemented. It may also be hard to control experiments with regards to repeatability and the effect of unknown variables, such as other users in the system and their network utilization. Real measurements also operate in real time, which means that it may take a long time to collect data, compared to a simulation running faster than real time. It depends on what is being measured and the research objective.

Simulation on the other hand can be fully repeatable and controlled, and may run faster than real time to allow for more variables or scenarios to be evaluated. Most of the papers surveyed in Section 3.1 have used simulations to evaluate the proposals put forward. However, simulation is not without problems. For example, [27] compares TCP measurements in a real test bed to the same scenario simulated with ns2 [157]. Even with a simple network topology, there are

[3]Parts of this section were published earlier in [5,81]. This is a revised version.

significant differences between the measured and simulated results. The difficulties of simulating the Internet is further argued in [74,76]. Further arguments on the choice of using simulation, live testing or emulation are available in [148]. The paper provides a good introduction to network emulation, architectural models and emulation approaches.

For our research, we use network emulation on the IP level. Network emulation has been used for quite some time in the networking community, and there exists a number of emulation packages. Dummynet [42, 188] is a network emulator originally designed for testing networking protocols. It simulates/enforces queue and bandwidth limitations, delays, packet losses, and network multipath effects (i.e., packet reordering via multiple pipes with different delays). Dummynet is integrated in FreeBSD and Mac OS X and is ported to Linux and Microsoft Windows. It can be used on user's workstations, or on dedicated PC's acting as routers or bridges. Pipes and queues are central concepts in Dummynet. By being integrated with the firewalling functionality, packets can be classified according to IP address, port number, type of service, and other fields. Matching packets can then be put into different pipes, where different pipes can have their own parameters for bandwidth limitations, delays, and so on. Pipes can also be entered probabilistically, so pipes with different delays can for example be used to introduce packet reordering. Dummynet, as shipped with the kernel, uses probabilities for packet loss. It uses a random source to decide which packets to drop, and can not introduce bit errors. These were required functionalities for our network experiments, to investigate the effect of packet loss position and the possible gains of handling segments with bit errors. As later described, Dummynet was chosen as the emulator basis, and extended with functionality to handle bit errors.

The AROMA test bed [149] is a real-time emulator for beyond-3G wireless networks. It is an ambitious effort, with a test bed architecture that includes the core network with routers and gateways, as well as the radio access network with signaling, transmission, radio signal propagation modeling. Real nodes can be connected to the end points to allow non-emulated applications and protocols to pass over

the emulation system. This test bed could probably be adapted to the Wireless IP system specifications and used for the experiments, but was not available at the time of our initial experiments.

NIST Net [168] allows a Linux PC to be set up as a router to emulate a wide variety of network conditions. These include tunable packet delay distributions, uniform or congestion-dependent packet loss, bandwidth limitation, packet reordering and packet duplication. With regard to packet loss, it can only be specified as probabilities and by statistical distribution. It can not introduce bit errors.

Seawind [126], the Software Emulator for Analyzing Wireless Data Transfers, is designed to study the data transfer for a single user. It can impose buffer constraints, bandwidth limitations, and a fine granularity of delays; allocation, transport, propagation and error delay can be set to closely emulate the delay behavior of a link. Seawind also features detailed logging. All arriving packets and events related to the packet is logged, and tcpdump-compatible output is produced. This allows for trace analysis with existing tools. The randomness used for packet loss in Seawind can be based upon various random distributions, but may also be given as a sequence in an external file. This may then allow for both repeatable and controlled loss.

Trace-driven approaches [160, 169, 170] measures real traffic and then use the information in the emulated scenario. In [160], probe packets are actively sent to another host, and the delay and loss is measured. In [169, 170], traffic is passively observed and distilled traces can then be used for example in the "PAcket Modulator" PaM [169]. PaM can drop, corrupt or delay intercepted packets, according to the network trace. Network traces does not directly allow for controlled loss, but an idea could be to create an artificial trace file that would cause errors at specific positions.

Other related tools includes ENDE [249], the End-to-end Network Delay Emulator, which allows for network experiments on a single PC. ONE [14], the Ohio Network Emulator, can be used to emulate a network between a pair of interfaces on a Solaris workstation. Delayline [104] differs from the other emulators in the respect that it operates in the applications themselves, rather than as a gateway. Ns-2 [157], the network simulator 2 and its sequel ns-3 [238] are

primarily discrete event simulation tools, but there exist functionality to integrate them into a live network.

The network experiments performed in this dissertation (Chapter 4 and Chapter 6) needed a network emulation with bit errors, as well as emulating/simulating detailed wireless channel and link properties. Most of the surveyed emulators do not introduce bit errors. Although Dummynet was lacking this functionality, it was found to be the most suitable emulator for developing bit error insertion[4]. The possible other candidates, Seawind, or trace-driven approaches, seemed overly complex, while Dummynet provided a good platform for further development. The bit-error extended functionality of Dummynet was used for experiments in Chapter 4. Further studies on the interaction between transport protocols and wireless networks required a more detailed emulation than what was reasonable in Dummynet, which prompted the development of WIPEMU described in Chapter 5. Although it is a stand alone emulator, it is using Dummynet for other tasks such as bandwidth limitation, delay and for diverting packets to be processed from the network stack to the emulator.

3.4 Summary

The problems for TCP/IP over wireless presented in the previous chapter has gotten a lot of attention in the research community. This is evident from the many optimization proposals presented in this chapter. Contrasting TCP to IPv4 and IPv6, there is no "TCP v2" being considered. Changes to TCP are continually being proposed and implemented to a varying degree, while still remaining compatible to the original TCP. The future extensibility of TCP is, however, starting to become questioned, as middleboxes modify passing data (for example network address translators that change port and sequence numbers, and transparent HTTP application proxies that removes TCP options) [98]. A general observation of the related work is that TCP/IP works well enough over most network

[4]This was based on an earlier implementation of bit errors by Jonas Rönngren and Kristofer Sandlund.

paths, although the described optimizations have shown that there is room for improvements. In this dissertation they are divided into four categories.

End-to-end proposals are based on the idea that complexity belongs in the end hosts rather than in the network. The end-to-end semantics of TCP are preserved. Intermediate nodes do not have to be TCP-aware, as in many of the other proposals. The data transferred between the end hosts is not required to pass the same intermediate node. Care should, however, be taken so that optimizations on one layer do not interfere with optimizations on another layer.

The link layer approaches mainly use link layer retransmissions to increase the reliability provided to TCP. The requirements on the link layer service may vary depending on the application. This is, for example, considered in UMTS, since the radio link protocol allows configuration of many parameters, such as the maximum number of retransmissions or amount of buffering.

Most of the explicit notification proposals require TCP-awareness of the intermediate node that is responsible for transmitting explicit notifications. These proposals seem to be intended for wireless links with low reliability. A link retransmission scheme with higher persistence would reduce or eliminate data loss due to an unreliable wireless link.

Split connection techniques use separate connections between the wired and wireless network. Performance improvements can be significant, but at the cost of violating the end-to-end semantics of TCP. The intermediate node which divides the connection must process data up to the transport layer and all TCP segments belonging to the same connection are required to pass that node.

Performance evaluations of live networks indicate that later 3G releases using HSPA have increased capacity and have a lower delay compared to earlier releases, and that TCP is able to achieve a high goodput. Live measurements give a more realistic performance expectation than for example simulation and emulation. However, if specific transport protocol issues are studied, a more controlled environment should be used, for example to remove uncertainty about the number of users in the system. To this end, early experiments

in this dissertation have used an extended version of the Dummynet network emulator. This software was used to perform bit-error related TCP experiments presented in the following chapter. The Dummynet modifications later evolved into the KauNet emulator [84,85]. To perform more detailed studies of the link layer, a new emulator WIPEMU was developed for the remaining experiments. WIPEMU and experiments are presented in Chapter 5 and Chapter 6.

Chapter 4

Accepting bit errors in TCP

Be conservative in what you do,
be liberal in what you accept from others.

— Jon Postel, RFC 761

The Internet has traditionally been used to transport content sensitive to errors. As it was designed for a wide range of heterogeneous networks, the transport protocol must take measures to ensure that the information is intact, even tough individual links in the network may be unreliable or cause packet corruption. It is intuitive that a corruption of executable content could cause it to malfunction at execution time. Even for non-executable content, such as an email message, data corruption could change the meaning of the message. On the other hand, there are applications that could tolerate corrupted data to an extent, or only require partial reliability. This chapter discusses the motivation and scope for such applications, and presents a modification to TCP to allow applications to make a trade-off between partial reliability (accepting corrupted data), and increased throughput. The modification is implemented in the Linux kernel, and the chapter concludes with experiments showing the potential performance gain compared to using a fully reliable TCP.

4.1 Motivation for partial reliability

The characteristics of the Internet user base has changed as the network has grown over the past decades. From being used mostly by military and academia until the mid 1990's, Internet is now in everyday common use. ITU reports that almost 70% of people in developed countries are Internet users [107]. One of the driving forces is entertainment, in the form of images, audio and video, termed multimedia content.

Multimedia content differs from traditional content (i.e., executables and emails) in a few ways. As it stimulates the auditory and visual senses, small errors in the presentation of sound or images can be accepted because the human perception regards the error as "noise", and compensates by interpolating from the other information perceived. Traditional content, for example a computer program, can not accept such errors because the computer assumes that all instructions are correct. Of course, an email message with spelling errors can probably be interpreted as the message the sender intended, but an unintentional corruption could happen at a place where it would go undetected.

Another difference is in the characteristics of data transport. A distinction can be made between bulk data transfer and "real time" data transfer. A bulk transfer is usually not sensitive to delays or jitter (differing inter-packet delays). The transfer rate may be allowed to fluctuate quite a lot, as long as a minimum rate is reached, with little impact on the application performance. Real time transfers, on the other hand, require that data are delivered on the expected time, or the data is not useful anymore. For example, a real time video stream consists of the transmission of a series of image frames. These must be played back at the same rate as they were transmitted for the user to be satisfied with the application. If a frame arrives later than its playback time slot, for example due to packet reordering, loss or delay, it can not be used any more since the video moment has already passed. The alternative would be to temporarily pause the playback and await packet reception. Commonly, a playback delay is used to keep a buffer of received data [132, Ch. 7.3].

Another area of consumer interest is mobility, as discussed in Chapter 2. Mobility is achieved by using wireless communication. The user has a mobile terminal, which communicates with a base station. The most common example is the cellular phone, but as the third and fourth generation mobile systems appear, the border between telephony and data services is being erased. Mobility over smaller areas can also be realized for example with WiFi hotspots. Besides the mobility aspect, 3G/4G wireless communication systems are rapidly replacing fixed broadband.

From these observations, it can be seen that multimedia content may accept errors, and has timing/throughput requirements on the transport characteristics for real time streaming. The question is therefore if these factors can be traded for better performance. Wireless radio networks is a problematic environment for Internet communications, where such a trade-off could be beneficiary. As discussed in Chapters 2 and 3, a wireless network exhibit higher error rates and greater delay variations compared to a wireline network, and there are many proposals to improve the TCP performance. These keep the paradigm of a fully reliable transport service. If instead partial reliability is acceptable, protocols such as UDP-Lite [136,137], Partially Reliable SCTP (PR-SCTP) [215], the Partially Reliable Transport Protocol (PRTP) [37], TCP-RC [158], Time-Lined TCP [165] and others [93] are examples where full reliability is traded for performance in some aspect.

With UDP-Lite, the UDP length field semantic is changed to instead mean checksum coverage. This means that the checksum can cover for example UDP/RTP headers, while allowing bit errors in the RTP payload. PR-SCTP allows for cumulative acknowledgements of missing data, upon agreement from both sender and receiver. PRTP allows controlled packet loss in a modified version of TCP, operating on the receiver side. The application can set a reliability level which indicates the amount of data loss the application can accept. A similar approach is used by TCP-RC, where the transport layer *forges* data to the application, if a segment is determined to be lost. The forged data is acknowledged to the sender, such that the sending rate should not be reduced. Time-lined TCP operates on the sender,

where the application sets deadlines for data transmission. If data cannot be transmitted within the deadline, for example due to congestion, it is removed from transmission buffers and replaced with more recent data.

The application must also be able to handle errors in the data stream. Examples of related work is [128] which describes a bit-error resilient packetization for streaming H.264/AVC video with UDP-Lite. For audio, [202] presents an algorithm to make MPEG AAC more resilient to transmission errors. For JPEG images, [82] shows a JPEG transcoder that handles packet loss (although corruption is not handled). With cross-layer information available to the application from the physical layer, [174] shows how bit errors may be corrected by using soft information from the channel reception to improve JPEG2000 decoding. Applications using uncompressed data, such as pulse-coded modulated audio (PCM/WAV [110] or bitmapped/pixelmapped images (BMP [77]) can also handle errors; a bit error in uncompressed audio results in a corresponding chirp when played back, and a bit error in an uncompressed bitmap image renders the corresponding pixel in error. The error is thus conveyed from the physical layer to the user, with a presumed user acceptance because the image is transferred more quickly. Errors may also be concealed by application filtering/smoothing to further reduce the impact of error.

A related technology is MAC-Lite [155], which is concerned with header estimation and recovery in WiFi, by using maximum a posteriori (MAP) estimation of the channel transmission. The intent is to provide header recovery for error tolerant protocols such as UDP-Lite, and simulations show an improved medium access header improvement of about 5 dB, compared to the standard decoder.

4.2 TCP-L overview

Building on the ideas of partial reliability, this section presents a technique for applications to receive data in TCP segments that have been corrupted by bit errors. Accepting and acknowledging, instead

of discarding, corrupted packets means that the TCP sender does not reduce the sending rate. Especially multimedia applications may benefit from trading bit errors to get better network performance[1]. For example, some pixels wrong in a picture will be compensated for by the human eye. Likewise for audio, the human ear will compensate for a few wrong sound samples. These applications could then benefit from an increased throughput and reduced jitter by doing a quality trade-off.

UDP is often used for transporting multimedia, depending on the needs of real time interactive communication. However, UDP lacks for example the congestion control features of TCP, and may therefore take an unfair amount of the available bandwidth, compared to TCP-friendly streams. This makes TCP an alternative for multimedia applications with soft real time constraints, such as streaming. TCP can be used for multimedia streaming by for example RealAudio [185] and Microsoft Windows Media Player [162]. Video streaming embedded on web sites often use Flash video (flv), where an Adobe Flash [3] application plays the video stream, delivered via TCP from the server. In the HTML5 standard being drafted [241], video playback is further integrated directly into the browser markup, without requiring external applications or plug-ins. The case for multimedia streaming with TCP is further argued in [87, 129].

To determine what gains can be made by accepting TCP segments with bit errors, a proof of concept of the idea has been implemented in the Linux kernel. This modification is called "TCP-L", as an indication of a more lightweight (fewer retransmissions) version of TCP. Parts of this chapter is published earlier in [6] and [39, Ch. 10].

To ease deployment, TCP-L only requires modification on the receiver side. Compared to modifying both the sender and receiver, it is easier for the end-user to deploy a solution which is not dependent upon the service provider to make changes to their systems. This approach can then be classified as an end-host modification, relating

[1] The underlying assumption is that the application using this modified TCP can handle errors, but this is subject to other research areas such as audio, image and video coding, as briefly discussed in the previous section. Also, the link layer must be able to deliver erroneous data to the transport layer of the terminal.

to the classifications discussed earlier in Chapter 3. The advantages of this class of modifications are analogous to the "end-to-end" arguments presented in [191]; end-points have the best knowledge of its communication, so that is where logic should be implemented.

4.3 TCP-L algorithm

The core idea behind TCP-L is to accept TCP segments with invalid checksums. However, since the checksum covers both the TCP header and the data payload, it is unknown if errors are in the header, payload or both. This separation is important because the header contains information to deliver the payload data to the correct application, and in the correct order. An option is to use a separate header checksum, as previously discussed in TCP-HACK [26] and CDO/CNO [239]. This is certainly possible, but it requires awareness on both the sender and receiver ends. One idea of TCP-L is to be a receiver-only modification. Therefore, to accept a segment with an invalid checksum, the issues with decoding and recovery of a possibly corrupt TCP header must be solved.

4.3.1 TCP header overview

The TCP header was shown and described in Chapter 2, but is reproduced here again in Figure 4.1 for easy reference. The header contains a number of control information fields, prepended to each payload data segment. The ports are used to determine which socket and application that data will be delivered to, while the sequence number determines where in the data stream the received data is placed. Further, the header contains control information (flags) used for connection establishment, closing, reset and other purposes. As seen in the figure, the header also contains a checksum field that is used to detect corrupted segments.

As earlier header compression research has shown, many fields in the header are constant or can be derived from other segments belonging to the same stream [59, 114]. The terminology *nochange*,

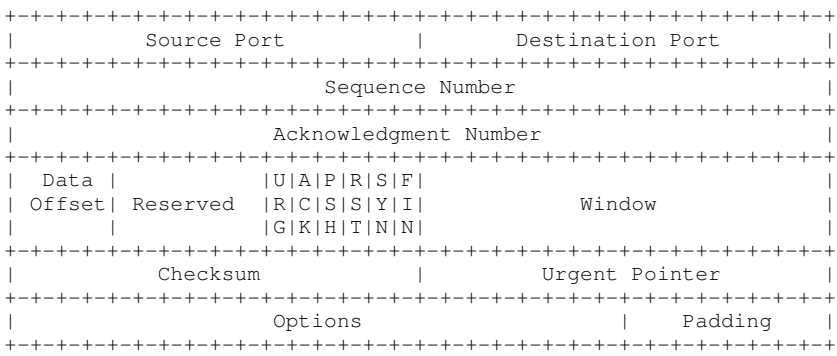


Figure 4.1: The TCP header (from RFC 793 [177])

inferred, *delta* and *random* is used in [59]. The terms refer to fields that are constant throughout the session, can be derived from other information, can be calculated from an earlier value plus a delta value, and fields whose content vary randomly, respectively.

We reuse the header definitions from [59] in Table 4.1[2]. It depicts the initial classification of the TCP header fields into the mentioned categories. The assumptions behind this classification are that every segment carries the same amount of data (to have delta-spaced sequence numbers). This is a reasonable assumption for constant-bitrate/full buffer media streaming. Further it is assumed that TCP options are not used, or at least that negotiated options are always present, to provide a known TCP header length.

nochange	source port, destination port, header length
delta	sequence number, ack number, adv. window
random	checksum, URG, ACK, PSH, RST, SYN, FIN, urgent pointer

Table 4.1: Classification of TCP header fields

In Table 4.1 there seems to be a lot of randomly varying fields which may be problematic to recover in the event of packet corruption. However, by making some assumptions, the table can be

[2]Except "inferred", because this is e.g. used for frame sizes derived from link layer information.

extended with an additional category, *dontcare*, and reconstructed as shown in Table 4.2.

nochange	source port, destination port, header length, URG, ACK, RST, SYN, FIN
delta	sequence number
random	
dontcare	ack number, adv. window, checksum, PSH, urgent pointer

Table 4.2: A re-classification of TCP header fields

The reasoning behind the additions to the *nochange* category is the following. The URG (urgent) flag is used mostly by terminal traffic type applications, i.e., not the multimedia application target. It is also the choice of the application to use this flag, so if an application enabled TCP-L it should also disable the use of URG functionality. TCP-L only recovers segments with non-empty data payloads. If a segment has an empty payload, it is of no use to the application. Further, if the checksum is incorrect, it means that all errors are in the header. Therefore it would be better to retransmit this segment than to try to recover it. This means that the SYN (synchronize) and FIN (finished) flags will always be set to zero, and the ACK (acknowledgement) flag always set to one [177] since the connection has been established and data is being communicated. When the RST (reset) flag is set, the segment contains no data. Therefore it is safe to always clear it if there is data in the payload.

The reasoning for the classification of the *dontcare* header fields is the following. The cumulative acknowledgement number in a corrupt segment can be safely replaced with the most recently correctly received acknowledgement number. Data is assumed to flow mainly in one direction, to the receiver, which acknowledges segments. This causes the sender advertised window to change minimally, as the sender does not get any data to fill its buffers[3]. The TCP check-

[3]Consideration has not been taken to other causes of advertised window changes, such as Freeze-TCP [89] or ERCN [234].

sum has already failed verification, so it is irrelevant to the processing. PSH (push) is a recommendation to the receiver to deliver any buffered data to the application, and it is reasonable to believe that an error in this flag would not cause harm. Since we assumed that no urgent data would be sent, the urgent pointer then becomes irrelevant as well. We further assumed that no options were used. If options still are used, recovery analysis for these must be done on an individual basis since content and purpose may vary.

The reconstructed classification makes it easier to handle errors in the header, since only the sequence number needs to be recovered. As long as the corresponding connection can be found, the other fields can be found or be set to safe values.

4.3.2 Phases of recovery

This section presents the steps that are taken in order to detect errors, locate connections and recover errors in the header. An overview is shown in Figure 4.2, and a discussion follows below.

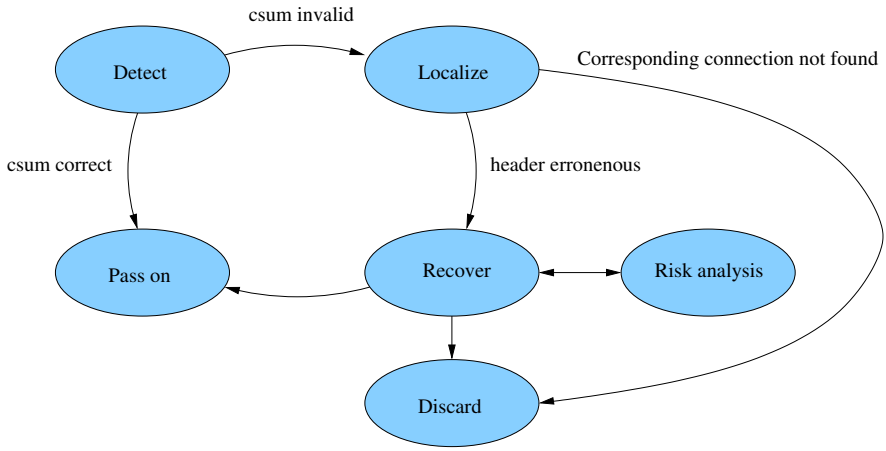


Figure 4.2: Recovery procedure overview

Detect The first step is to detect that an error has occurred. This is normally done by verifying the checksum of the segment, but may

also be determined from the link layer, depending on the possibilities of information exchange between the layers. If the checksum is correct, the segment is passed on normally. Otherwise, the segment is processed to localize the corresponding TCP connection.

Localize When a segment has been determined to contain errors, the corresponding connection for the segment must be found. This is necessary to make use of constant and temporal information for the connection, such as ports and previous sequence numbers. This localization can be carried out with help of the header characteristics discussed in Section 4.3.1. This is done for example by trying to match port and sequence numbers to the active connections in the system. The search for a corresponding connection is further narrowed to connections where TCP-L is enabled. However, too much effort should not be spent trying to find a match. If large parts of the header has been damaged, it is probable that the payload is also severely damaged and less useful to the application.

Risk analysis If a header field does not correspond to an expected value, there are two options. Either the value is kept, or a presumably better value is chosen. This choosing implies a risk, and the risk analysis must consider the consequences of changing a given header.

The recovery algorithm works by first matching the segment to an existing connection. This is a critical step, because choosing the wrong connection may cause data to be injected into a completely foreign stream. If there are few streams to different hosts and varying port numbers, the chances of mapping a segment to the wrong connection is minimized. However, if there is a set of connections that are connected to the same end point and have consecutive local port numbers, the risk of mapping the segment to the wrong connection is increased.

Another critical issue is the sequence number. This determines where in the stream the payload belongs. How critical depends on the applications. Can it tolerate “data reordering”, or “data truncation” for example? These considerations should be decided by the application, since it has to be able to handle the delivered data.

Recover In this step, the segment has been determined to contain errors, and if they are in the header they should be recovered. The detailed algorithm is described in Section 4.3.3 below. After the header has been recovered, the segment is passed on as if the checksum was initially correct.

Discard In the recovery phase, chances are that the header could not be satisfactorily recovered. For example, when doing the risk analysis a change may be deemed too risky to perform, but also too risky not to perform. One solution that can always be used in these cases is to simply discard the segment. This is what happens normally when the checksum is invalid, and causes a retransmission of the segment.

4.3.3 Recovery algorithm

Given the assumptions outlined earlier with regard to the header field properties, a possible way to implement the recovery phases is outlined below:

1. A corrupt TCP segment is detected by an invalid checksum.

2. The segment is mapped to an existing connection, identified by {*src addr, dest addr, src port, dest port*}. If no exact match is found, an approximate matching is tried. That is, removing for example *src port* from the matching or doing bit-level matching. If several connections are found to be possible candidates, sequence and acknowledgement numbers are used to choose the most probable connection.

   If no connection is found to be a close enough match, the segment is discarded.

3. If the constant header fields (Table 4.2) are incorrect, they are changed to match those of the found connection. The "dont-care" fields are set to safe defaults.

4. The sequence number is checked to see if it is the expected one, in relation to earlier received segments. If it is not expected, there are three causes.

   (a) The sequence number is corrupt

   (b) The segment has arrived out of sequence

   (c) The segment belongs to another connection

   These causes could also be combined, i.e., a wrong segment out of order has a corrupt sequence number.

   If the received sequence number equals the last sent acknowledgement, we assume it is correct. From the assumption that the sequence number is delta spaced, multiples of segment sizes can be added to the last sent acknowledgement to see if it matches. Then, the segment may be assumed to be delivered out of order. However, if the sequence number does not meet the criteria above, the segment should be discarded. This is to minimize the risk of placing data at the wrong position in the data stream, which may be harder to handle for the application compared to bit errors.

   There is, however, a risk that a reordered segment with bit errors match the expected sequence number. Consider a simple example with segment sizes of 256 bytes. Segment four with sequence number 1024 is expected but a segment with sequence number 1280 arrives. In this case, a single bit error could cause the sequence number 1280 (binary `10100000000`) to be classified as sequence number 1024 (binary `10000000000`). Having both reordering and such unfortunate bit error is considered unlikely. Also, the segment size in the example was a worst-case chosen for illustration. Standard segment sizes of 1460 bytes would be even more unlikely to exhibit this misclassification.

4.4 Experiments

The recovery algorithm above is implemented in the Linux kernel, with the limitation that no recovery attempt is performed if port numbers and sequence numbers do not match an existing flow (segment is discarded). This is to minimize the risk of misplacing data. As the experiment results show, there is not much room for further performance improvement so no further optimization of the algorithm was implemented.

The implementation was used in experiments aimed to compare the performance of regular TCP to TCP-L, from a protocol point of view. The performance was believed beforehand to be improved, but the degree of improvement would be shown through experiments.

The physical experiment setup consisted of three networked computers (Figure 4.3). One sender, one gateway which emulated the wireless link and introduced errors in passing segments, and one receiver containing the TCP-L modification. The network emulator used, Dummynet with bit error insertion, is partly described in [81], and also mentioned in Section 3.3.

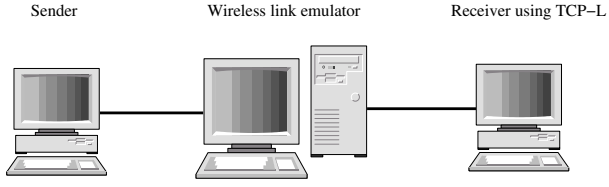


Figure 4.3: Experiment setup

The link was set to emulate basic characteristics of a 3G/UMTS wireless transmission with a 384 kbit/s bandwidth and 70 ms delay, with a varying amount of residual bit errors applied to packets passing the gateway. The experiment consisted of sending 1 MByte of data from the sender to the receiver. The receiver application enabled or disabled TCP-L in the transport layer. Each experiment run was replicated 30 times to obtain statistic confidence intervals [117]. Network traffic was captured with the `tcpdump` tool at the receiver, and this was then analyzed with the `tcptrace` [171] tool to calculate

the throughput and provide data for plotting graphs[4]. The results of this analysis is summarized in the next section.

4.5 Experiment results

The experiment used two different error models. First, the bit errors were randomly distributed with an error probability varying from 0 to $2.5 * 10^{-5}$. The second error model used a fixed bit error rate at $1 * 10^{-5}$, but with different error distributions.

4.5.1 Randomly distributed errors

Figures 4.4 and 4.5 show a comparison between TCP and TCP-L in the presence of varying amounts of randomly distributed errors. Two different maximum transmission units (MTU) of 576 and 1500 bytes per packet are used. The throughput where no errors are applied indicate the upper bound of the throughput capacity for the link configuration and packet size, and corresponds to the theoretical throughput. As more and more errors are introduced, more packets become corrupted. TCP, the lower curve in both graphs, discards the corrupted packets, reacts to the packet loss and lowers the sending rate quickly, since the assumption made by TCP is that packet loss is due to a congested network, as discussed earlier in Chapter 2. TCP-L, the upper curve, is able to recover from most of the errors. As the error rate increases, fewer packets can be recovered, leading to a minor throughput reduction, compared to the reduction of regular TCP.

Another aspect that can be seen from the two figures is the impact of packet size. In Figure 4.4, a packet size of 576 bytes is used. This means that there is more overhead for segment headers, which leads to a reduced optimal throughput. As the error rate increases, the smaller packet size starts to compare more favorably for TCP. There is less risk of a packet being in error compared to the larger packet

[4]This basic setup, data collection and analysis is also used for experiments in later chapters.

size in Figure 4.5. At BER $2.5 * 10^{-5}$ and MTU 576, TCP reaches slightly more than 10 kbyte/s. For MTU 1500, slightly less than 10 kbyte/s is reached. Also, for the smaller packet size less bandwidth is wasted when retransmissions occur, compared to retransmissions with the larger packet size. The effects of packet size on throughput are, however, very minor, compared to the effect of congestion control.

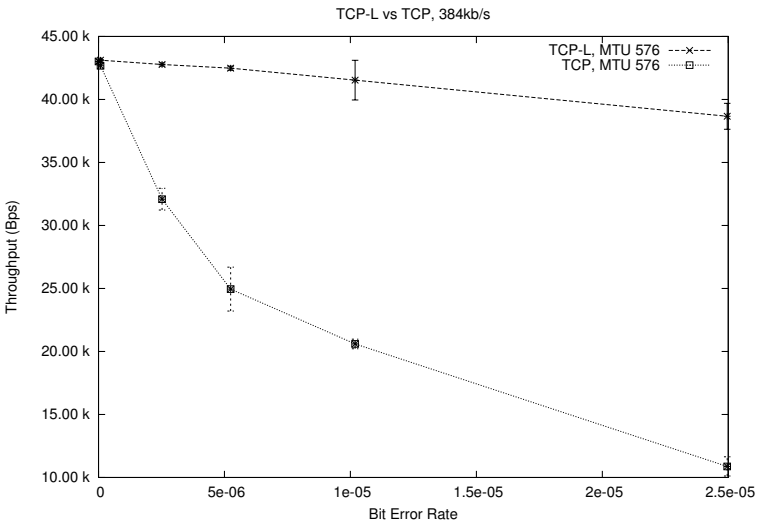


Figure 4.4: Increasing amount of randomly distributed errors, MTU 576

4.5.2 Impact of burstiness

The above experiments assumed that all errors were randomly distributed. Depending on the link technology, errors may appear in bursts. The impact of error burstiness was explored by using a fixed amount of errors at 10^{-5}, and then varying the distribution of bit errors. This was implemented by using Dummynet emulation extended with bit error patterns. A fixed amount of n bit errors were placed at different positions in the loss pattern according to a two-state markov model (see Figure 4.6) with a "random position" and a "burst position" state. State transitions depend on the burstiness

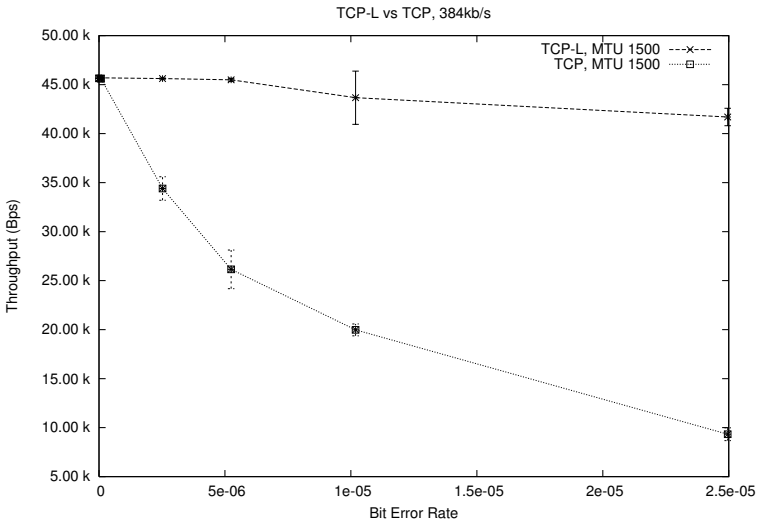


Figure 4.5: Increasing amount of randomly distributed errors, MTU 1500

factor β, $0 \leq \beta \leq 1$. While in the "burst placement" state, the bit error is placed within ± 8 bits distance from the previous bit error. For example, with $\beta = 0$ all bit errors are randomly placed, while $\beta = 0.5$ means there is a 0.5 probability that a bit is placed randomly or nearby the previously placed error.

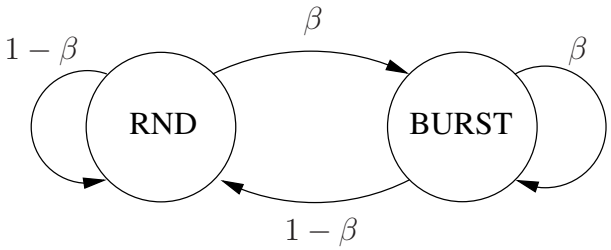


Figure 4.6: Markov model used for bit error bursts. In the RND state, bit errors are placed randomly. In the BURST state bit errors are placed nearby the previously placed bit error.

Figures 4.7 and 4.8 show the results from these experiments. The x-axis shows the probability that errors occur nearby each other. At zero percent, the bits are randomly distributed. This result corre-

sponds well with the previous experiments at the same bit error rate (10^{-5}). As the burst probability increases, the rate of error burstiness increases as bit errors become more and more clustered.

In Figure 4.7 the graph shows that as burstiness increases, throughput for TCP is improved. The reason for this is that there is no difference if there is one or more errors in a packet, it will be discarded anyway. But with a fixed amount of errors, and if these errors group together, there will be fewer packets overall that are corrupted. A lower packet loss rate results in higher throughput performance, as seen in the presented experiments.

TCP-L maintains a throughput near the optimal, independent of the burst distribution. A small improvement is made as bit errors become more and more grouped together. Before the experiments were done, it was unclear how error burstiness would affect the performance of TCP-L. The main question was if a few longer error bursts would make error recovery harder, compared to recovery from single bit errors. But, as seen in the figures, TCP-L also improves its throughput when there are longer bursts. Even though damaged packets become increasingly harder to recover, they also become fewer, which upholds the throughput.

The experiments in Figure 4.8 uses the larger packet size. As for randomly distributed errors, the impact of lower overhead for larger packets is seen.

4.6 Conclusions

This chapter investigated the idea of accepting corrupt TCP segments to improve the throughput of TCP in a wireless environment for applications that may tolerate a partially reliable transport service. From the presented experiments we conclude that network performance can be upheld, depending on link and error characteristics, compared to using a standard TCP that decreases throughput because of perceived network congestion. For this to be possible, some conditions must be met. The link layer should deliver erroneous data to the transport layer (i.e., GSM with a disabled reliable link pro-

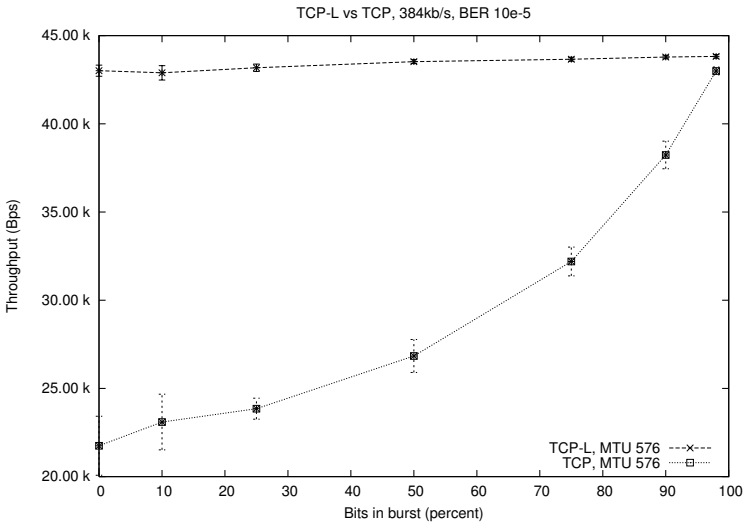


Figure 4.7: Increasing error burstiness, fixed BER. MTU 576

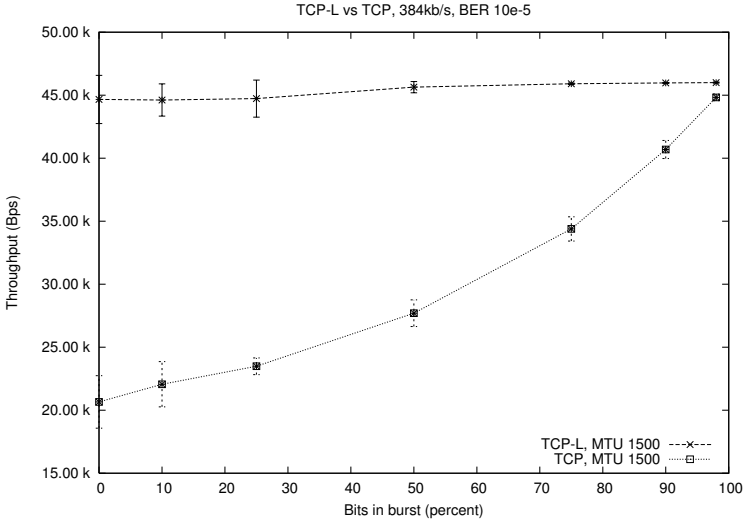


Figure 4.8: Increasing error burstiness, fixed BER. MTU 1500

tocol). The application should also be able to handle errors, and to communicate to TCP-L when errors are acceptable or not. The application must also be able to handle the risk of misclassification

of corrupted data, where a corrupted packet destined for another application erroneously is mapped to the TCP-L connection. This is a drawback and consequence of using a receiver-only modification, but the risk must nevertheless be taken into account. To minimize the risk of injecting data in the wrong connection, only packets with port and sequence numbers matching an existing connection should be recovered.

On a further note, the performance improvements have not taken additional link retransmissions into consideration. This means that the amount of residual bit errors are reduced and often eliminated. Related work have shown the benefits of such, and it is also shown in Chapter 6. Generally if the link retransmission delay is short compared to the total end-to-end delay, local retransmissions should be used. For links with higher retransmission delay, such as GSM/ WCDMA or satellite transmission, this would provide better opportunities for TCP-L.

Chapter 5

The Wireless IP evaluation system

This continuing race of increasing sequence numbers of mobile systems is in fact just a matter of labels. What is important is the actual system capabilities and how they have evolved.

— Dahlman, Parkvall, Sköld [55]

Communication technology is continuously evolving. The trend among the different wireless technology generations discussed in Chapter 2 is pointing toward higher capacity with larger coverage. There is a rapid increase in devices with multiple radio interfaces, and consumer cell phones are available with GSM, 3G, WiFi and Bluetooth all integrated. These technologies serve different bandwidth requirements at different ranges of coverage. With an increased interest in higher data rates on longer distances in combination with mobility, there is a need for new radio interfaces. Several technologies have been developed for this purpose, such as 3G high speed packet access (HSPA), 3GPP Long Term Evolution (LTE) and WiMAX. Later systems have moved from using CDMA to OFDM as the transmission technology.

This chapter describes an earlier approach, the Wireless IP downlink evaluation system, which combines high data rates and vehicular speeds. Further, an emulator of this system is presented and

validated. The chapter ends with an overview of system emulation parameters that are used for the experiments in the following chapter.

5.1 Wireless IP

One 4G[1] system proposal based on OFDM has been developed within the "Wireless IP" project [212] at Uppsala University, in collaboration with Chalmers University of Technology and Karlstad University during the years 2002-2008. As mentioned in Chapter 2, input from this project was carried to the WINNER project, which in turn influenced the currently commercially available 3GPP LTE system.

The main focus of Wireless IP is to cover wide areas to service stationary as well as vehicular users in excess of speeds of 100 km/h with a 30-fold bandwidth increase compared to UMTS/3G. To realize this goal, adaptive OFDM is used in combination with channel prediction. Transmissions can then be scheduled to maximize the total satisfaction of the users, depending on their current channel quality. This is combined with increased cross layer interaction, link level ARQ, and other mechanisms.

The Wireless IP evaluation system is planned as an all-IP system intended for Internet traffic. It is designed to provide a good service to the network and transport layers and it is thus important to consider the system level implications of lower layer design decisions and settings. In order to allow performance measurements on the interaction between the physical/link layer design and upper layers an emulator, Wireless IP EMUlator (WIPEMU), was developed to emulate the Wireless IP downlink system proposal.

5.2 Downlink system proposal

The infrastructure of the Wireless IP evaluation system [10, 211, 220] consists of base stations with sectored antennas and mobile terminals.

[1] In this dissertation the term "4G" is used in more general terms and not specifically for 3G LTE, LTE-Advanced or ITU IMT-Advanced.

The available downlink bandwidth within a sector is slotted in time. Each time slot is further divided in frequency, forming time × frequency *bins*. The system proposal assumes a bandwidth of 5 MHz, using an OFDM modulation of each bin using 20 subcarriers of 10 kHz each, for in total 200 kHz per bin. This is appropriate for vehicular and stationary terminals in urban or suburban environments [235].

With a 5 MHz bandwidth, this design results in 25 available bins in each time slot, as illustrated in Figure 5.1. Scheduling granularity operates on a bin-basis, such that individual bins can be scheduled for specific terminals. For frequency-selective fading channels where different terminals experience different fading, this allows for a higher system capacity compared to a design that would allocate full time slot(s) (TDMA) or frequency slot(s) (FDMA) to an individual terminal.

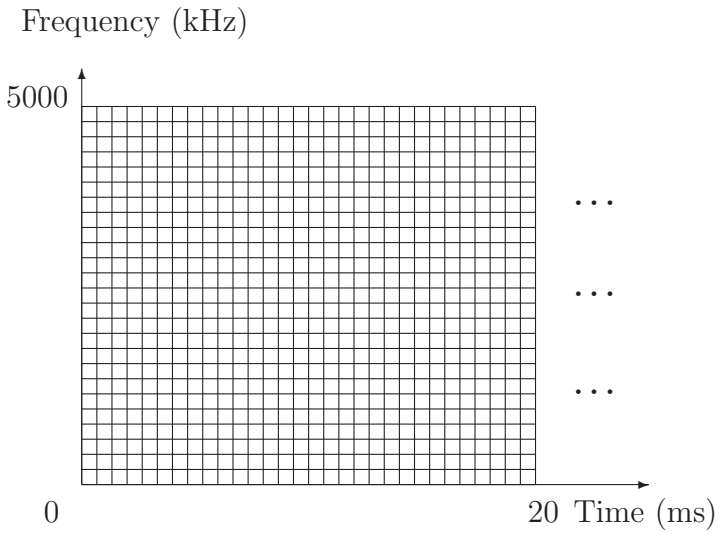


Figure 5.1: The structure of a 5 MHz band, divided into 25 bins ×200 *kHz for a duration of 30 time slots* × 0.667 *ms, corresponding to* 20 *ms.*

The details of a bin are shown in Figure 5.2. Each bin carries $20 * 6 = 120$ symbols in 0.667 ms × 200 kHz. Of these symbols,

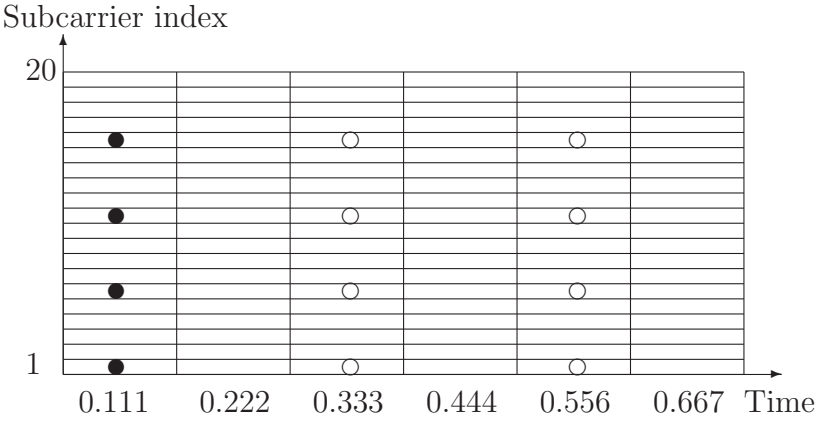


Figure 5.2: The bin structure [211]. 20×6 *symbols are modulated over the time interval (*$T = 0.667$ *ms) and each subcarrier (*20×10 *kHz), in total 120 symbols. Of these are 4 known pilot symbols (black dots) and 8 downlink control symbols (circles) leaving 108 symbols for the payload (blank).*

12 are used as pilot symbols for training and control, and 108 are available for data transmission [209].

Table 5.1 shows the transmission strategy. During time slot $j+0$, the terminal performs a channel prediction of the signal to noise ratio (SNR) for all bins for a prediction horizon corresponding to the retransmission delay. This information is sent to the base station during time slot $j+1$ (during which the channel again is predicted for the next transmission). The base station performs scheduling among all terminals during time slot $j+2$, and then transmits during time slot $j+3$. Therefore, three time slots are required from signaling to reception of the requested data, resulting in a delay of $3 * 0.667 =$ 2 ms. The signaling include possible requests for retransmission of bins that were received corrupted, and the minimum retransmission delay is therefore 2 ms.

The 108 payload symbols are modulated using an adaptive modulation system, where the applied modulation is chosen based on the predicted channel on the terminal. An uncoded modulation system

Slot	j+0	j+1	j+2	j+3
Time (ms)	0.667	0.667	0.667	0.667
$j = 0$	Predict SNR (T)	Signal (T)	Schedule (B)	Transmit (B)
$j = 1$	...	Predict SNR (T)	Signal (T)	...
$j = 2$	...	...	Predict SNR (T)	...

Table 5.1: Slot usage from the terminal side. The terminal (T) first predicts the channel, and then signals the prediction to the base station (B) during the following time slot. The base station schedules the transmission during the third time slot and finally transmits during the fourth time slot.

is used with 8 levels, with modulation index $M = 2^k$ for k bits per symbol, using BPSK, 4-QAM, 8-CrQAM, 16-QAM, 32-CrQAM, 64-QAM, 128-CrQAM and 256 QAM. The channel is assumed to be sufficiently flat during the transmission that all payload symbols in a bin use the same modulation.

An important aspect of an adaptive modulation system is the estimates from the channel prediction that is used to choose the modulation index. Details and surveys on channel prediction are available in [19, 63, 209], and here we focus on the accuracy of the prediction. Channel predictors use measurements of known pilot symbols and knowledge of the statistical properties of the channel variations, to provide a prediction of the channel SNR at some point in the future. The accuracy of the prediction varies with prediction horizon, carrier frequency, fading environment, pilot symbol density, and velocity of the terminal. Intuitively, longer prediction ranges, higher carrier frequencies, fewer pilot symbols, and higher terminal velocities contribute to a less accurate prediction.

Beside the point estimate of the future channel power gain, the predictor must provide a measure of the prediction accuracy. To quantify this we use the normalized mean squared error (NMSE), of the signal to noise ratio, defined in Eq. (5.1) [210].

$$\text{NMSE} = \frac{E_h|\gamma_{u,l} - \hat{\gamma}_{u,l}|^2}{E_h|\gamma_{u,l}|^2} \tag{5.1}$$

Here, E_h is the estimate of the channel h, with instantaneous SNR γ for user u and bin l, and the predicted instantaneous SNR $\hat{\gamma}$.

The NMSE range investigated in this dissertation is 0-0.2, with a design target NMSE of 0.1 for reasonable vehicular velocities and prediction horizon of three time slots ($\lambda = 0.333$ wave lengths) operating at a 2 GHz carrier frequency. The SNR predictions and corresponding accuracy estimates are used to select the modulation index for a transmission. The SNR limits for the modulation indexes can be adjusted depending on different criteria, such as optimizing for a target bit error rate, or to maximize the amount of bits in correctly received payloads. The latter strategy is used in this dissertation to provide maximum throughput in combination with persistent retransmissions, while taking the prediction accuracy into account [210].

Channel quality predictions can also be used during scheduling. In a multi-user environment, different users experience different channel quality. One user may have a good channel at a specific bin, while another user has a bad channel at the same bin. The scheduler can then optimize the system throughput by scheduling the user with the best predicted channel quality. Two simple scheduling strategies are used in this dissertation: one that operates in "FDMA"-mode, without consideration of the predicted channel quality. The same bin indexes are used for all transmissions. The second assigns the n bins with the highest SNR to the user.

5.3 WIPEMU

WIPEMU is a software emulator implementing the functionality of the Wireless IP downlink system proposal, mainly developed by the dissertation author as part of the Wireless IP project. The purpose of the emulator is to investigate how various settings or design decisions on the link layer would impact the performance on the transport or application layer. The emulator functions as an IP gateway through which traffic is routed. Passing packets are picked up by the emulator and subjected to a simulated wireless transmission. Therefore, WIPEMU can be plugged into a real network environment, enabling a wide range of operating systems and network stack implementations to be tested, since the interface is a regular Ethernet connection

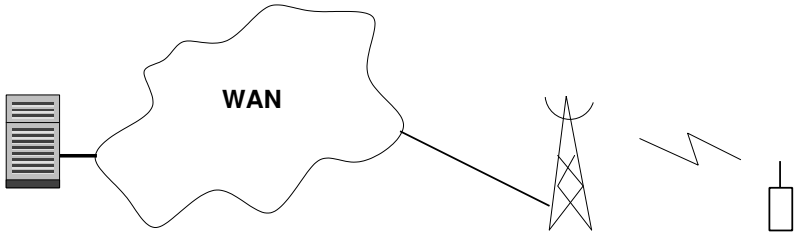


Figure 5.3: The emulated network scenario. A wireless terminal connected via a base station, connected via a wide-area wired network, is communicating with a server.

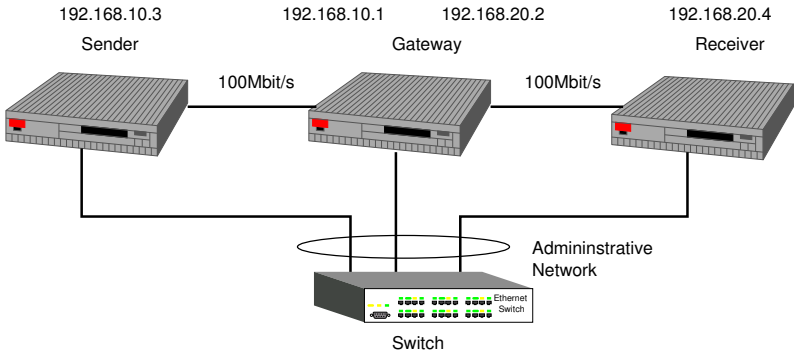


Figure 5.4: The hardware setup that implements the emulated scenario.

and IP packets are processed. An early presentation of WIPEMU is published in [7], and this section contains an extended and revised description.

5.3.1 Overview

The network scenario of a typical emulation is shown in Figure 5.3. Here a server is connected to a wide-area network (e.g. the Internet), which in turn is connected to a base station, that serves the terminal of a mobile user. The user can then use different communication patterns such as file transfer or real-time voice communication. The scenario is emulated with a three-computer setup shown in Figure 5.4. One machine acts as the server (sender), a second machine is a gateway (WIPEMU), and the third machine is the mobile user

(receiver terminal).

The overall functional design of WIPEMU is to collect IP packets arriving at the gateway, process these according to the downlink specification, and then transmit the processed packets to the receiver. Performance measurement can then be done at multiple points: in the applications that communicate, or on the IP level of the sender and receiver. Performance data and statistics are also produced by the emulator itself, such as how many frames have been used in each modulation level, or the amount of frame retransmissions.

The internal functionality of WIPEMU is separated into three parts executing as concurrent threads via the pthread library: the *collector*, the *timer* and the *simulator*. These are combined into a single executable in FreeBSD [86]. The *collector* utilizes Dummynet [42,188] and packet diversion that is part of the FreeBSD firewall functionality. First Dummynet can be used to emulate various conditions of the wired network such as bandwidth and delay limitations, or packet loss. The divert socket is then used to move packets from kernel space to user space, where the emulator is running. The *timer* function is responsible for keeping a coarse-grained synchronization. It consists of a coarse-grained outer synchronization loop that sleeps the major part of a 10 ms time interval and then busy-waits until the end of the time interval. It then notifies the simulator thread that it should start processing packets in the transmission queue. The simulator thread is then responsible for the fine-grained synchronization on the per-time slot basis. After the simulator has processed 15 time slots, a flag is set to indicate finished execution. The outer loop then verifies that the emulator is synchronized, and then performs the next simulator invocation.

5.3.2 Simulator implementation details

The *simulator* thread is where the main processing is implemented. Every 10 ms it is triggered by the timer, and starts the processing of packets in the transmission or retransmission queue. Since the slot duration is 0.667 ms, this means that 15 time slots are processed every invocation.

Within each time slot there are 25 bins available for transmission, corresponding to a total bandwidth of 5.0 MHz. Each bin has a corresponding channel condition expressed in SNR. Each bin also has a predicted SNR. If adaptive modulation is used, the predicted SNR is used to perform a table lookup and choose a suitable modulation level.

The modulation level impacts the amount of data carried in each bin. For example, a bin with 108 symbols modulated with 4-QAM would carry 108 $symbols * 2\ bits/symbol = 216\ bits = 27\ bytes$. This amount of data is then extracted from the transmission queue and put into a link frame. The simulation thus operates on a bin/symbol granularity. OFDM processing, channel estimation and prediction is abstracted in the SNR averaged for each bin.

The symbols in the link frame are "transmitted", by generating complex random numbers from a normal distribution with a standard deviation depending on the non-predicted SNR for the bin. This corresponds to the addition of white Gaussian noise to the transmission. Every symbol is then "detected" by verifying if the complex random number is inside or outside the detection region of the transmitted symbol. If a "received symbol" is outside the detection region, the frame is marked as failed. The failed frame is then put in a retransmission queue with an added retransmission delay. Retransmissions take precedence over data in the transmission queue. The maximum amount of retransmissions before a frame is considered failed can also be limited. A setting decides if an IP packet with failed frames should be dropped or be delivered with the residual bit errors. This way an upper bound of the packet delay can be determined, although correct packet delivery is not guaranteed. An overview of the transmission emulation is shown in Figure 5.5.

When a frame is retransmitted, the channel conditions may have changed such that another modulation level should be used. For example, a frame might be transmitted with 4-QAM and fail. When the frame is retransmitted, a BPSK modulation may be the most suitable. Clearly all the data in the original transmission does not fit. In this case the original transmission is split up into two frames. The reverse may also happen that a retransmission experiences better

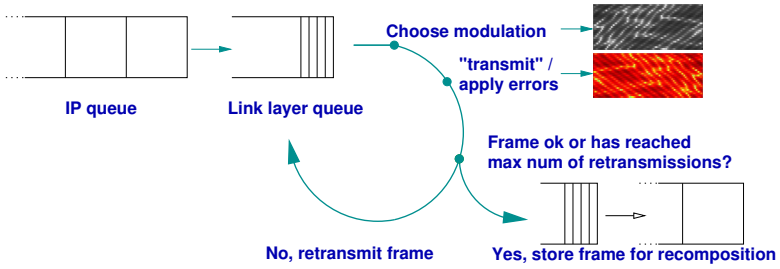


Figure 5.5: The WIPEMU core functionality

channel conditions and could use a higher modulation. In the simulator we have chosen to keep the original modulation to keep complexity down. The alternative would to be increase the modulation to pack more data in the frame. If the extra data is not contiguous with the transmitted data it would require extra control information inside the frame, possibly eliminating the extra efficiency gained. Another aspect is that the symbol error probability is reduced by using a lower modulation than what is optimal for a given SNR, increasing the probability that the frame is retransmitted correctly.

To reduce the amount of retransmissions, a chase combining [50] function is implemented. When a frame transmission has been received with errors, the complex valued symbols of the frame are stored. When a retransmission of the same data and same modulation arrives, the two frames are combined to reduce the noise component in the transmission and increase the likelihood for successful decoding. This is further discussed in Section 6.2.2.

Timing and synchronization was previously mentioned. Each 10 ms the simulation is triggered. However, it is also important to be synchronized within the time interval. The transmission simulation runs faster than real time (otherwise the simulator would not be able to keep up), which would mean that perhaps only 4-5 ms of real time would be used in the 10 ms interval. From an external standpoint, this would lead to packets being delivered earlier than in a real system. In addition, IP packets arriving in the later part of the interval would not be processed until the start of the next interval. To solve this problem, the simulation function itself keeps

a virtual time that is incremented for each 0.667 ms transmission time interval. After all bins in one transmission time interval are processed, the simulator busy-waits until the real time has reached the virtual time. Therefore, IP packets (arriving e.g. 6 ms after the interval start) are processed accordingly at the correct time even when the system is idle and would have finished processing very early.

5.3.3 Validation of WIPEMU

A validation of the emulator has been performed to see how the theoretical throughput and measured throughput match. A close match indicates that the emulator functions properly and does not drift, i.e., does not send data too slow or too fast as compared to what it should. The evaluation has been done by comparing the theoretically obtainable throughput to the throughput observed in experiments, while compensating for protocol overhead. This validation is also partly published in [9].

The WIP downlink system proposal is configured to transmit up to 25 bins 1500 times per second. With each bin containing 108 symbols, and each symbol carrying 1 to 8 bits, the maximum theoretical throughput T_t can be calculated with the formula in Eq. (5.2).

$$T_t = i * f * s * n \tag{5.2}$$

Here, i is the number of time intervals per second (1500), f is the number of bins per time interval (25), and s is the number of symbols per bin (108). Finally n is the number of bits per symbol. The calculated values for full utilization, with 25 bins per time unit with 1 to 8 bits per symbol, are shown in Table 5.2.

bits/sym:	1	2	3	4	5	6	7	8
kbit/s:	4050	8100	12150	16200	20250	24300	28350	32400

Table 5.2: Maximum theoretical channel throughput (kbits per second), using 25 bins per time slot, at 1-8 bits per symbol, assuming error free transmission, for 5 MHz bandwidth using the Wireless IP downlink proposal.

The spectral efficiency is a common measure of the performance of a digital communications system and is specified in bits per second per hertz. This ranges between 0.81 (1 bit/symbol) to 6.48 (8 bits/symbol), for the payload data transmission (excluding the 12 symbols used for modulation training and control).

The number of bits per symbol depends upon the modulation strategy. This can either be fixed at a certain level throughout the transmission, or be adapted to the current channel condition.

Fixed modulation

The throughput with fixed modulation is determined by performing experiments where the modulation level is not changed during the transmission. To be able to correlate the measured throughput we assume an error free transmission, i.e., no retransmissions at this stage.

The measured throughput is calculated with the tcptrace [171] tool, operating on tcpdump packet captures at the receiver. This throughput is thus calculated on the application layer. To compare this to the theoretical throughput, the TCP/IP header overhead must be compensated for. In addition, the last part of the IP packet does not typically fill an entire link layer frame, which also needs to be compensated for in the comparison.

The IP packets are 1500 bytes, with a 52 byte header. The network throughput, including headers, is therefore $52/1448 = 3.59\%$ higher than the application throughput. The frame truncation depends on the link modulation level, and is calculated according to Eq. (5.3).

$$\frac{bits\_in\_frame - bits\_of\_data\_in\_last\_frame}{total\_number\_of\_bits\_per\_packet} \tag{5.3}$$

For the Wireless IP system and IP packets of 1500 bytes, this overhead is given by Eq. (5.4), where M is the modulation index.

$$\frac{108 * log_2\ M - 1500 * 8\ mod\ (108 * log_2\ M)}{1500 * 8} \tag{5.4}$$

Tables 5.3–5.5 compare the theoretical and measured throughput based on the discussion above, for 1, 13 and 25 bins per time slot. The tables show the number of bits carried in each symbol per modulation level (1), and the theoretical maximum number of bits per second (2). The tables show further the measured throughput on the application level (3), and the throughput used by headers (4) and compensation for truncation in the last frame (5). Comparing columns (6) and (2) it is seen that the measured throughput with compensation (6) and the theoretical throughput (2) match closely, as shown in column (7).

For minimum and half system capacity (tables 5.3 – 5.4, the theoretical and measured result match in the order of 99.9%. For full system capacity (table 5.5) the results also match very well, in the order of 99.7 to 99.9%, which is slightly less than for the lower capacities. This is due to an increased sensitivity in the result because of measuring on the application layer, for example due to the TCP slow start at connection initiation. These issues existed also in the measurements for fewer bins per time slot, but were less pronounced due to the lower throughput.

The conclusion of the tests is that the theoretical and measured results is a very good match, and validates the function and performance of the emulator in fixed modulation mode.

Adaptive modulation

Validation of the adaptive modulation is more complex to perform than that of fixed modulation. This is because the channel changes continuously, which results in a varying amount of data being transmitted with each bin, making it harder to determine the unused capacity (overhead) in the last bin of an IP packet.

Because of this, a special channel was constructed for this validation step. Each time interval consists of a sequence of SNR-values that match the switch interval of each of the eight modulation levels: $\{8, 12, 15, 18, 22, 25, 28, 32,\ 8, 12, 15, 18, 22, 25, 28, 32,\ 8, 12, 15, 18, 22, 25, 28, 32, 0\}$ dB. These correspond to the 25 bins in each time interval (bin 25 with 0 dB is not used).

Within each time interval, the first bin has a SNR of 8 dB that

(1) bits/ sym	(2) theoretical (kbits/s)	(3) measured (kbits/s)	(4) header (kbits/s)	(5) trunc (kbits/s)	(6) total (kbits/s)	(7) total/ theoretical
1	162	156.4	4.28	1.25	161.91	0.9995
2	324	312.8	8.57	2.50	323.83	0.9995
3	486	460.9	12.63	11.98	485.53	0.9990
4	648	625.5	17.14	5.00	647.63	0.9994
5	810	761.5	20.86	26.65	809.03	0.9988
6	972	921.7	25.25	23.96	970.87	0.9988
7	1134	1094.5	29.99	8.76	1133.25	0.9993
8	1296	1250.6	34.26	10.00	1294.88	0.9991

Table 5.3: Comparison of theoretical and measured throughput when using fixed modulation formats, without packet errors. One bin per time slot is used (minimum system capacity).

(1) bits/ sym	(2) theoretical (kbits/s)	(3) measured (kbits/s)	(4) header (kbits/s)	(5) trunc (kbits/s)	(6) total (kbits/s)	(7) total/ theoretical
1	2106	2032.6	55.69	16.26	2104.56	0.9993
2	4212	4064.9	111.37	32.52	4208.75	0.9992
3	6318	5989.6	164.10	155.73	6309.47	0.9986
4	8424	8128.0	222.68	65.02	8415.68	0.9990
5	10530	9892.1	271.02	346.22	10509.34	0.9980
6	12636	11975.3	328.09	311.36	12614.75	0.9983
7	14742	14218.2	389.54	113.75	14721.50	0.9986
8	16848	16249.9	445.20	130.00	16825.15	0.9986

Table 5.4: Comparison of theoretical and measured throughput when using fixed modulation formats, without packet errors. 13 bins per time slot are used (half system capacity).

(1) bits/ sym	(2) theoretical (kbits/s)	(3) measured (kbits/s)	(4) header (kbits/s)	(5) trunc (kbits/s)	(6) total (kbits/s)	(7) total/ theoretical
1	4050	3908.6	107.09	31.27	4046.96	0.9992
2	8100	7815.6	214.13	62.52	8092.24	0.9990
3	12150	11514.8	315.47	299.38	12129.66	0.9983
4	16200	15609.1	427.65	124.87	16161.66	0.9976
5	20250	19001.9	520.60	665.06	20187.52	0.9969
6	24300	23002.5	630.21	598.07	24230.82	0.9972
7	28350	27333.4	748.86	218.67	28300.90	0.9983
8	32400	31235.2	855.76	249.88	32340.84	0.9982

Table 5.5: Comparison of theoretical and measured throughput when using fixed modulation formats, without packet errors. 25 bins per time slot are used (full system capacity).

causes the adaptive modulation to select 1 bit/symbol modulation, the second bin has a SNR of 12 dB that would use 2 bits/symbol, and so on. This results in the first bin being transmitted carrying $1\ bit/sym * 108\ sym/bin = 108\ bits$, the second bin transmitting $2 * 108 = 216\ bits$, the third bin $3 * 108 = 324\ bits$, and so on. For $n \in 1, 2, 3$ multiples of 8 bins in a time slot (8, 16, and 24 bins), this corresponds to $n * (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8) * 108 = n * 3888$ bits per time slot.

The amount of truncation in the last frame varies with the amount of used bins, since an IP packet does not align with an even number of 8-bin-multiples. An IP packet contains $1500 * 8 = 12000$ bits, which requires $12000/3888 = 3$ 8-bin-multiples with 336 bits remaining. If the transmission started at 1 bit/symbol, these 336 remaining bits will be transmitted in one bin of 108 bits, one bin of 216 bits and one bin of 324 bits. The last bin will not be fully utilized, causing a truncation overhead of $108 + 216 + 324 - 336 = 312$ bits. This means that the second IP packet will start by transmitting a bin of $4 * 108 = 436$ bits. The amount of remaining bits will vary, as will the amount of truncation.

An approximation of the truncation overhead is therefore calculated by observing the distribution of packet boundaries over the different modulation levels, assuming continuous transmission with queued packets:

1. The first IP packet starts using the bin at 1 bit/symbol, then 2 bits/symbol, ... until 24 bins are used. As discussed, there are 336 bits left, 2 more complete bins are therefore required (1, 2 bits/symbol), and a third incomplete bin (3 bits/symbol) is required that results in the truncation overhead.

2. The second IP packet thus starts using the bin at 4 bits/symbol. Like the first IP packet, there are 336 bits left after 24 bins are used. However, due to the bin offset, these 336 bits are contained in one additional bin of 4 bits/symbol ($4 * 108 = 432\ bits$).

3. The third IP packet then starts using the bin at 5 bits/symbol,

Starting bits/sym	Complete bins	Bits left	Last bin size (bits)	Truncation overhead (bits)
1	8+8+8+2	12	324	312
2 (infreq)	8+8+8+1	120	324	204
3 (infreq)	8+8+8+1	12	432	420
4	8+8+8	336	432	96
5	8+8+8	336	540	204
6	8+8+8	336	648	312
7	8+8+8	336	756	420
8	8+8+8	336	864	528

Table 5.6: Truncation overhead for adaptive modulation

after 24 bins there are 336 bits left, which are contained in one additional bin of 5 bits/symbol.

4. The fourth, fifth and sixth IP packets starts using the bins at 6, 7, 8 bits/symbol and operates as in step 2-3 above.

5. The seventh IP packet starts using the bin at 1 bit/symbol, and the cycle is repeated from step 1.

This means that packets are evenly distributed over 6 out of 8 modulation levels and start at 1, 4, 5, 6, 7, 8 bits/symbol. In the event that a transmission would start at 2 or 3 bits per symbol, for example if short packets are processed (SYN, FIN, etc), the transmission becomes synchronized with the 1,4,5,6,7,8 transmission when full sized packets are sent again.

Table 5.6 shows the overhead depending on the starting modulation level. This corresponds to a mean truncation overhead of $(312 + 96 + 204 + 312 + 420 + 528)/6 = 312$ bits for every IP packet, corresponding to $312/(1448 * 8) \approx 2.69\%$ overhead on an IP packet (excluding 2 and 3 bits/symbol starting offsets). As with the fixed modulation, there is a TCP/IP header overhead of 3.59% that must be accounted for when comparing the theoretical throughput to the application throughput.

(1) bins/time slot	(2) theoretical (kbits/s)	(3) measured (kbits/s)	(4) header (kbits/s)	(5) trunc (kbits/s)	(6) total (kbits/s)	(7) total/ theoretical
8	5832	5486.3	197.0	147.6	5830.9	0.9998
16	11664	109722.8	393.9	295.2	11661.9	0.9998
24	17496	16372.6	587.8	440.4	17400.8	0.9946

Table 5.7: Comparison of theoretical and measured throughput when using adaptive modulation, without packet errors, for varying system capacity.

The results of the above discussion are summarized in Table 5.7. The table shows the system loaded at about 1/3, 2/3 and 3/3 capacity. As for the fixed modulation, the theoretical and measured throughput match closely (99.98%), with a small deviation for the highest load (99.46%).

Validation conclusion

Based on the validation of the fixed and adaptive modulation above, it has been shown that the core function of the emulator is matching the theoretical expectations and is able to keep up with the full system load.

5.3.4 Parametrization

There are a number of parameters available for adjustment in the emulation system. Table 5.8 contains a compilation of the relevant parameters, their supported value ranges and default settings. The parameters are grouped into the wired network, wireless downlink and wireless uplink. The wired network parameters are set in the Dummynet network emulation. The wireless downlink parameters are set in the WIPEMU emulator at the emulation start. The wireless uplink is not part of the WIPEMU emulation and is emulated with Dummynet. Settings marked as “unlimited” in the table means that there is no upper limit for all practical purposes. The parameter details and cross-layer performance impact is discussed and analyzed in detail in the following chapter.

5.4 Summary

This chapter presented the background of the Wireless IP downlink evaluation system, the design and implementation of an emulator of the system, a performance validation of the emulator and a emulator parameter overview. The WIP system is specified down to the OFDM transmission with subcarriers and inter-symbol spacing. However, the emulation granularity is on a link frame/bin level. This is because the radio channel simulation is too computationally demanding to be carried out in real time. Instead, the radio channels are simulated beforehand to produce SNR values for each bin as input. This data is then used by the emulator to determine appropriate modulation levels and error probabilities.

Parameter	Supported value range	Default setting
Wired network		
Wired network delay (ms)	0 ms – unlimited	10 ms (RTT=20 ms)
Network queue size (packets)	1 – 100 packets	54 packets
Packet loss probability	0 – 1	0 (uncongested)
Wireless Downlink		
Frame transmission time (ms)	0.667 ms	0.667 ms
Frame retransmission delay (ms)	2 ms – unlimited	2 ms
Channel models	12-tap Jakes typical urban Winner C2 typical urban Ericsson Research channel measurements	*All channels are used*
Modulation	Fixed or adaptive modulation, 1-8 bits/symbol	Adaptive modulation
Frame size (symbols)	108 payload symbols	108
Coding	Uncoded M-QAM	Uncoded M-QAM
Scheduling	Static or best predicted channel allocation	Best predicted channel allocation
Link ARQ policy	0-unlimited with or without soft combining	Max 30 retransmissions, with soft combining
Handling of corrupt frames/packets	discard or deliver	deliver
Wireless uplink		
Channel model	Capacity, delay and packet loss can be specified	
Capacity (bits/s)	0 – 100 Mbit/s	500 kbit/s
Delay (ms)	0 ms – unlimited	Included in wired network delay
Packet loss probability	0 – 1	0

Table 5.8: Emulator parameters

Chapter 6

WIPEMU experiment results

A man with one watch knows what time it is;
a man with two watches is never quite sure.

— Segal's Law

The Wireless IP downlink system was described in the previous chapter. Numerous analytical evaluations [208] have been performed by other researchers in the project to show the advantages of the system design. Of interest here is the system performance with real applications and network traffic. This chapter presents the results from experiments with the WIPEMU network emulator. The chapter starts with a description of how experiments are executed, along with parameters that are common to all experiments. The chapter continues with investigation of the baseline parameter settings such as adaptive modulation, link retransmission policy, and basic channel allocation. Based on this a set of default settings are determined. From this baseline, other parameters are varied to study the impact of link retransmission delay, delay in the wired network, loss in the wired network, and buffer sizing. Further two network applications with diverse requirements on the network transport characteristics are studied to investigate the interaction between settings on the link layer and the result on the application. The chapter ends with the experiment conclusions.

6.1 Common experiment settings

There are a number of settings that are common between the experiments, while other settings are unique. These common settings are discussed in this section, while the unique settings are discussed together with the respective experiment details.

First, the experiment setup and execution is described. Then follows a description of the wireless channel data and related issues.

6.1.1 Experiment execution

The experiment setup used is the same as used in the previous chapter on emulator validation. One machine acts sender, sending data through a gateway that runs the WIPEMU emulator, which arrive at a third machine acting as the receiver.

Most experiments analyze the achieved TCP throughput at the receiver, while isolating the impact of the studied parameter settings. A single user and a single TCP flow[1] are used to be able to isolate the specific effect of the parameter being studied.

For each parameter setting, the netperf [119] application is used to establish a TCP session, transmit data during 30 seconds, and close the connection. This is repeated 30 times with a different channel realization from the same channel model (see description below). During each TCP session, raw packet data is captured at the receiver with the tcpdump application. After the repetitions, the parameter settings are advanced and another run begins.

When all runs are finished, the resulting packet captures are analyzed with tcptrace [171] to extract the throughput. From each experiment run the data is collected and the mean and confidence intervals are calculated from the 30 data points. The data is then plotted in the respective graphs.

Some analysis includes metrics other than TCP throughput, such as TCP retransmissions, link frame retransmissions, or distribution

[1] In a real environment there would be multiple users and multiple flows, but their interaction would make it harder to determine if a certain throughput is due to the number of users, competing flows, or the parameter being studied.

of the modulation index. This data is produced using the experiment execution as described, but data is collected from other parts of the tcptrace analysis, or from emulator generated output.

The above execution is used in Sections 6.2 and 6.3, while Section 6.4 uses a varied TCP flow length, as well as UDP traffic (but still for a single user with a single flow).

For the experiment results, a notation of A, B, C and a number is present at the top of the graphs. These can be used to correlate the experiment with the specific parameter set in the experiment plan, see Appendix A.

6.1.2 Channel models

The channel model represents the radio signal variation over time, and is central to the function of the emulator. It is used to both choose an appropriate modulation level for transmission, and to determine successful or erroneous reception.

Three different radio environments are used for the channel models, to strengthen the generality of the results. These are named "Jakes", "Winner" and "Measured". The first two consist of synthetic/simulated data, while the third consists of channel sounding measurements in an urban area. As mentioned in the previous section, each channel model is simulated or measured 30 times to obtain statistical confidence intervals [117, Ch. 13]. When the experiments are compiled, the mean value and 95% confidence interval of the measurements over the 30 channel realizations are presented. The three channel models are further described below.

Jakes

The "Jakes" channel model [118] is a multipath Rayleigh fading channel with a "typical urban" profile, with the user moving at 75 km/h, with 12 power delay profile taps, at [−5.7, −7.6, − inf, −4.4428, −13.4, − inf, −13.5793, −14.2371, −14.4794, −16.9543, −20.0164, −24.3], at 1800 MHz carrier frequency and 5 MHz bandwidth. It also includes a slow/shadow fading component, from an autoregres-

sive AR(1) process with a variance of 4 dB and pole at 0.74, giving a shadow length of about 2 meters.

Winner

The "Winner" channel model is based on the propagation scenario C2 NLOS, typical urban macro-cell metropolitan, from the IST WINNER deliverable 5.2 [159], with the user speed range of 0-70 km/h, and a 3000 m cell size. The carrier frequency was 1900 MHz, with a 5 MHz bandwidth for transmission.

Measured

The "Measured" channel model is based on channel sounding measurements performed by Ericsson Research in Kista, Sweden [19, Ch. 8.2]. The measurement was made with a car moving at 30 km/h, with a carrier frequency of 2660 MHz, 19.44 MHz bandwidth and using four MIMO antennas. From this channel data, a 5 MHz band was extracted to match the bandwidth of the other channels.

6.1.3 Channel prediction

The adaptive modulation discussed later depends on having access to predicted channel data as a basis for choosing an appropriate modulation level [210]. In a real system, this data is continuously provided by a channel predictor. For performance reasons of the emulation, such predicted channel data has been calculated in advance. For each of the 3*30 channel realizations discussed above, there exists a set of four "predicted" channel realizations with a varying prediction error, as well as a "perfectly predicted" channel realization which is identical to the channel itself.

The channel prediction errors are NMSE 0, 0.0085, 0.0356, 0.0963 and 0.20, corresponding to the use of prediction horizons of 0, 0.083, 0.167, 0.250 and 0.333 wavelengths, respectively. These prediction horizons (λ_{pred}) were chosen based on an 1800 MHz carrier frequency (f_c), with varying amount of prediction horizon in time ($t_{pred} = timeslots * 0.667\ ms/timeslot$) or varying mobile velocity ($v_u(m/s)$),

according to Eq. (6.1) where c is the speed of light, approximated to $3 * 10^8\ m/s$.

$$\lambda_{pred} = \frac{t_{pred} * v_u * f_c}{c} \tag{6.1}$$

The prediction errors used correspond to a 75 km/h velocity and 0, 1, 2, 3, and 4 time slots prediction horizon. With a three timeslots prediction horizon and varying velocity, they also correspond to 0, 25, 50, 75, and 100 km/h. A change in carrier frequency could also be recalculated according to the formula. The actual cause for the varying prediction error may therefore be due to a shorter or longer prediction horizon, varying user velocity or varying carrier frequency. The underlying cause of the prediction is not considered, but the experiments observe the results of having different prediction errors.

6.1.4 Channel utilization

In a multi-user setting, there are multiple users sharing the available resources according to some criteria such as application bandwidth or delay requirements, or economic incentives (i.e., premium service fee). Thus having all bins allocated to only one user would be unrealistic, as well as only having very few bins available for every user (implying an over-subscribed network). In the experiments, a value of 10 bins per timeslot was chosen, corresponding to a $10/25 = 40\%$ utilization, to get a utilization between the two extremes. This is for a bandwidth of 5 MHz, which is also what is used in UMTS W-CDMA.

6.1.5 Mobility model

A mobility model corresponds to the pattern of movement and changing velocity of a mobile user in a landscape. This gives rise to fading, interference, and other phenomena that affect the channel conditions. In our experiments there is no explicit mobility model defined. The channel conditions are instead generated from the mathematical channel models described aboveor from channel measurements.

6.2 Baseline experiments

There are five basic design aspects that are investigated in this section. They involve the use of fixed versus adaptive modulation, the amount of link retransmission attempts, the use of symbol combining, modulation switch level optimization and the channel allocation. In the following sections we discuss the details of each aspect, present results from experiments and conclude with a set of recommended settings. These results are also partly published in [7, 8] and [9].

6.2.1 Fixed vs. adaptive modulation vs. ARQ

A fundamental design choice is how data should be modulated. Should a fixed modulation or adaptive modulation scheme be used? The idea is that using adaptive modulation better utilizes the fluctuations in channel quality and can (per definition) adapt the transmission to such changes [219]. Another fundamental choice is that of frame retransmissions. Should retransmissions be used? How many retransmission attempts should be performed in that case?

Modulation strategy

Data is transported within a bin (also called link layer frame or, in LTE systems, a resource block). A bin of the modeled transmission system contains 108 symbols for data transport. All symbols within a bin use the same modulation, from BPSK to 256-QAM. This corresponds to 1 to 8 bits per symbol, or 108 to 864 bits/bin. The modulation can be fixed for all bins, or it can be adapted for every bin according to predicted channel conditions. This adaptation is based on a switch table that defines the SNR intervals for each modulation level [210]. The switch levels in the table can be optimized with certain criteria. For example depending on the channel prediction error, a target bit error rate or to achieve maximum throughput. A modulation level is chosen for the bin that is transmitted, based upon a channel prediction and a corresponding modulation level according to the switch table. The accuracy of the channel SNR prediction,

the chosen modulation level and the true channel SNR impacts the symbol detection and the resulting bit error rate.

A common technique to reduce the bit error rate is to use coding [203], which in combination with retransmission is called Hybrid ARQ [144]. Coding adds extra information to the transmission and can be used to recover data that has been corrupted. The extent of recovery depends on the type of coding and the amount of extra information added. A code that transforms m bits of information (data bits) into n coded bits is denoted an m/n rate code. For example, a 1/2 (half) rate code contains two coded bits for every data bit, while a 5/6 rate code contains 6 coded bits for 5 data bits.

The effect of coding is a reduction of the bit error rate compared to an uncoded transmission, for the same channel SNR, essentially shifting the bit error curve a few decibels. In this dissertation only uncoded transmission has been used, due to the complexity of implementation and the number of other parameters to investigate. The effect of coding is however believed to be similar to that of using uncoded transmission with a higher mean SNR in the experiments [207]. Paper [70] shows 1-2 dB gain in spectral efficiency by using adaptive Trellis-coded modulation (TCM) compared to using uncoded M-QAM, for a large range of average SNRs. In cases of high average SNR, M-QAM outperforms TCM. In paper [67], uncoded and coded adaptive OFDM modulation is compared. Coded modulation is found to reduce the bit error rate, but it also reduces the throughput due to the coding overhead.

ARQ/Retransmission

Given the inherent fluctuations in radio channel quality, compared to for example fiber optic transmission, a main question is how failed transmissions should be handled. Since transport layer packets are broken down into multiple link layer frames, a failed frame transmission can be handled in a few possible ways. Either the transport layer packet that contains the frame can be discarded (dropped), or delivered with the erroneous frame. If TCP is used this usually results in an end-to-end transport level retransmission of the whole

packet, and the possible invocation of congestion control. If UDP is used, the response to the lost packet depends on the application. Another option is to retransmit the failed frame over the local link. This would instead add delay to the packet delay on the transport layer, but should give a better performance than doing an end-to-end retransmission.

Two aspects of the retransmission mechanism are further investigated: the retransmission persistence and retransmission delay. The persistence determines how many retransmission attempts should be performed before giving up and declaring the frames as failed. A low persistence means a higher packet loss rate (leading to a lower TCP throughput), but it can also serve to impose a limit on the total transport packet delay. Depending on the data transport characteristics of the application, such as real time media streaming, delay may be more important than packet loss. However, if the retransmission delay is low, more retransmissions can be done while keeping an "acceptable" delay compared to if the retransmission delay is higher. The delay impact is further investigated in Section 6.3.1.

Experiment results

Figures 6.1 to 6.3 show the experiment results of using fixed versus adaptive modulation versus ARQ persistency for the three channels, with the experiment setup and execution as described earlier. The TCP throughput is plotted as a function of the maximum allowed link frame retransmissions. The different graphs represent adaptive modulation and various levels of fixed modulation.

The key result shown in Figure 6.1 for the Jakes channel is that adaptive modulation outperforms fixed modulation. The problem with fixed modulation is that it either does not use the available capacity efficiently, i.e. a higher modulation could have been used, or that the modulation is too high, resulting in a high bit error rate. For example BPSK performs well even with few link retransmissions. This is due to a low bit error rate for this combination of modulation level and channel capacity. Doubling the modulation to 4-QAM almost doubles the throughput, but this modulation level needs to

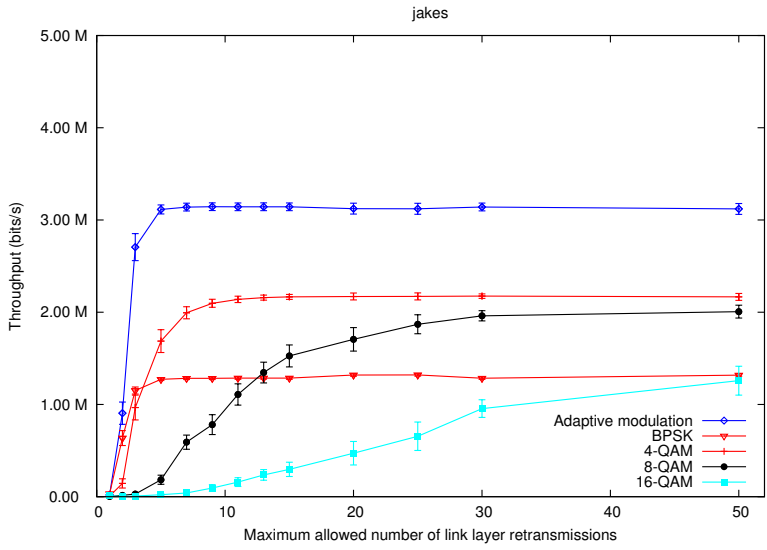


Figure 6.1: TCP throughput of the Jakes channel, for different modulation schemes, and varying maximum number of link frame retransmissions. 5 MHz bandwidth at 16 dB mean SNR.

do more link retransmissions. If the fixed modulation level is increased further, this results in a lower TCP throughput due to the fact that more and more link frames experience failure and require retransmission.

Adaptive modulation on the other hand uses the (predictably) best modulation for every transmission. There are failed frames requiring retransmissions, but it is still a better strategy compared to using fixed modulation[2], throughput wise. As seen in the figures, adaptive modulation reaches an asymptotic maximum throughput after about 5 maximum retransmissions. Most frames are, however, delivered after two retransmissions, as shown in Fig. 6.7(a) in a later section.

Figures 6.2 and 6.3 show the results for the Winner and Measured

[2]It should be noted that no power control has been taken into account, which can be used to increase the signal strength. On the other hand this can lead to problems with interference with neighboring transmitters.

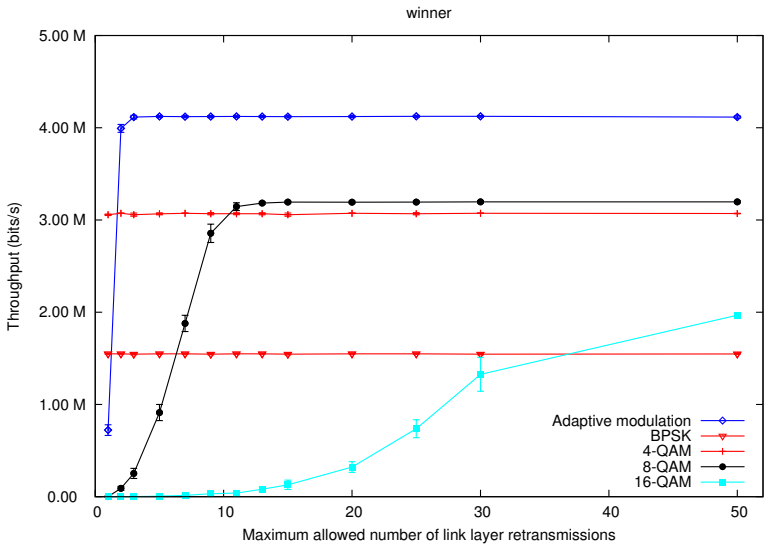


Figure 6.2: TCP throughput of the Winner channel, for different modulation schemes, and varying maximum number of link frame retransmissions. 5 MHz bandwidth at 16 dB mean SNR.

channel, which show the same characteristics as the Jakes channel. One notable difference from the Jakes channel is that 8-QAM is now the fixed modulation level that gives the highest TCP throughput when many link retransmissions are allowed. This is because the channels have different properties, and with many retransmissions more frames are transmitted successfully with the particular modulation order. However, the main result still holds that adaptive modulation is the best choice, and it reaches an asymptotic throughput with only a few retransmissions.

6.2.2 Symbol combining

In the experiments so far, each received frame has been individually processed. Frames in error have been discarded and retransmitted. However, the frame can still be useful when processing retransmissions of the same frame at the same modulation level. By combining the symbols of one or more retransmissions, symbol detection with

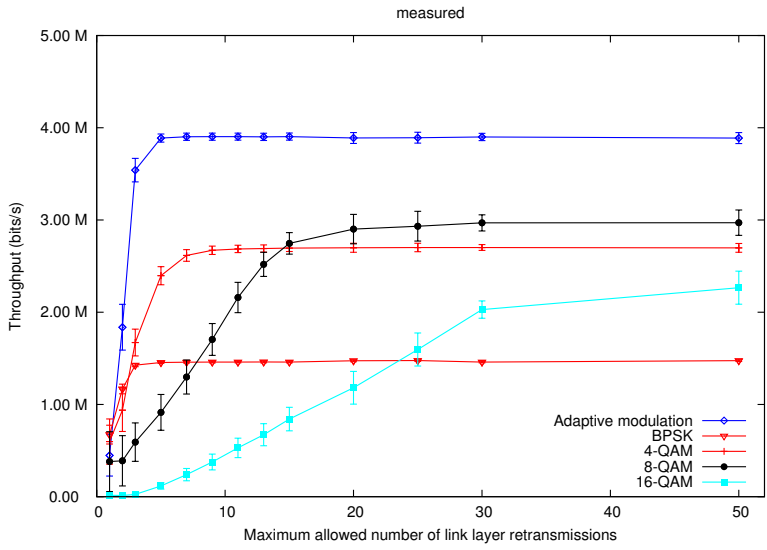


Figure 6.3: TCP throughput of the Measured channel, for different modulation schemes, and varying maximum number of link frame retransmissions. 5 MHz bandwidth at 16 dB mean SNR.

lower error probability can be achieved compared to detecting symbols with no prior information.

In detail, after a symbol is modulated and transmitted, it is detected and decoded at the receiver. Depending on the amount of noise in the received signal, there is a varying probability of symbol error. This probability can be estimated analytically (see for example [179, Ch 5] and [90, Ch 6]) to calculate an approximation of the symbol or bit error rate, based on the modulation level and SNR. To get more realistic error emulation, the emulator instead performs a detection based on the symbol constellation and the SNR.

The symbol detection is implemented by generating a complex number from a Gaussian distribution for every symbol, with a standard deviation determined by the SNR according to the channel data for the current bin being processed. The resulting constellation point is then compared to the region of the transmitted symbol constellation to determine if it is inside or outside the detection region.

If a symbol is incorrect (outside its detection region), the whole

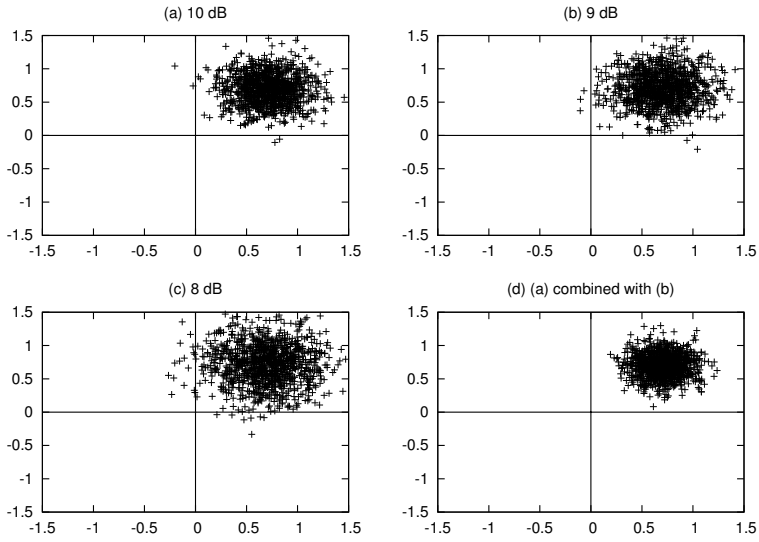


Figure 6.4: Example of symbol combining. Plot (a), (b) and (c) show 1080 4-QAM modulated symbols with an AWGN noise of 10, 9, and 8 dB, respectively. All three plots have points crossing the axis to another quadrant which become symbol errors. Plot (d) shows instead a combination of (a) and (b) that has no symbol errors, which illustrates the reduced need for retransmissions.

frame is marked as being in error, and is handled by the retransmission mechanism. In the case of erroneous transmission, the detection features a chase combining [50] method to increase the likelihood of successful detection. If a bin is received with symbol errors, the symbols are kept and combined with the retransmitted symbols, by taking the complex mean weighted by the respective SNR. Figure 6.4 illustrates this effect. It contains four subfigures with a constellation diagram of 4-QAM, where 1080 symbols (10 frames of 108 symbols) are transmitted and subject to an additive white Gaussian noise. In plot (a), having a 10 dB SNR, most symbols are contained in the upper-right quadrant (constellation point $\{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\}$) and therefore correctly decode to the transmitted bits. However, some symbols have crossed the axis and would be decoded incorrectly (belonging to another constellation point), resulting in a symbol error and leading

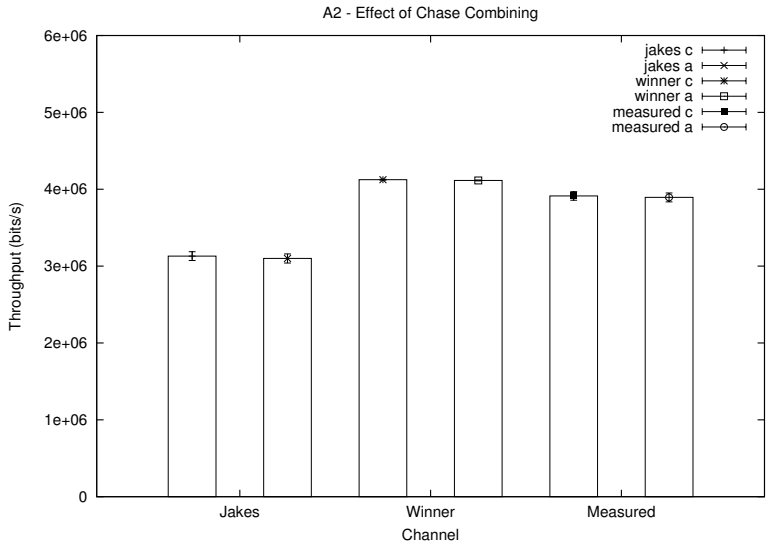


Figure 6.5: TCP throughput of the Jakes, Winner, and Measured channels with chase combining (labeled "c") and without combining (labeled "a").

to bit error(s). The frames containing such a symbol would therefore be retransmitted. Plot (b) represents retransmissions at a 9 dB SNR, where even more symbols are incorrectly decoded, and another retransmission is needed: plot (c) is a third retransmission at 8 dB SNR, which also fails and would require a fourth retransmission. However, transmission (a) and (b) can be combined (plot (d)), and all symbols are now decoded correctly.

The weighted chase algorithm considers two complex symbols $\mathbb{Y}_1$ with SNR γ_1 and its retransmission $\mathbb{Y}_2$ with SNR γ_2 which are combined into symbol $\mathbb{Z} = \mathbb{Y}_1 * \frac{\gamma_1}{\gamma_1+\gamma_2} + \mathbb{Y}_2 * \frac{\gamma_2}{\gamma_1+\gamma_2}$. Should more retransmissions be required, these retransmitted symbols are added to the previously combined symbols while recalculating the weights of all included symbols.

Experiment results

Figure 6.5 shows the throughput difference between using combining or not. From the graph it is difficult to make out if there is a difference

or not, or if it is outside the confidence intervals. Therefore, we study the amount of link retransmissions, shown in Figure 6.6.

Like the throughput, the retransmissions also show no major improvement when combining is used. However, when doing even further investigation of the link frame retransmission distribution, Fig. 6.7(a), there is an intelligible difference with non-overlapping confidence intervals when combining is used. Figure 6.7(b) is in logarithmic scale to better show the impact of each retransmission. The reason why symbol combining does not have a greater effect is because combination is only applied to retransmissions. For example, with a 10% frame retransmission rate, 90% of the transmitted frames are received successfully. To elaborate, the probability of successful reception with one transmission and one retransmission is $1 - (1 - 0.9) * (1 - 0.9) = 0.99$. Therefore 1% of the frames require more than one retransmission. Combining the first transmission and the subsequent retransmission further reduces the need for multiple

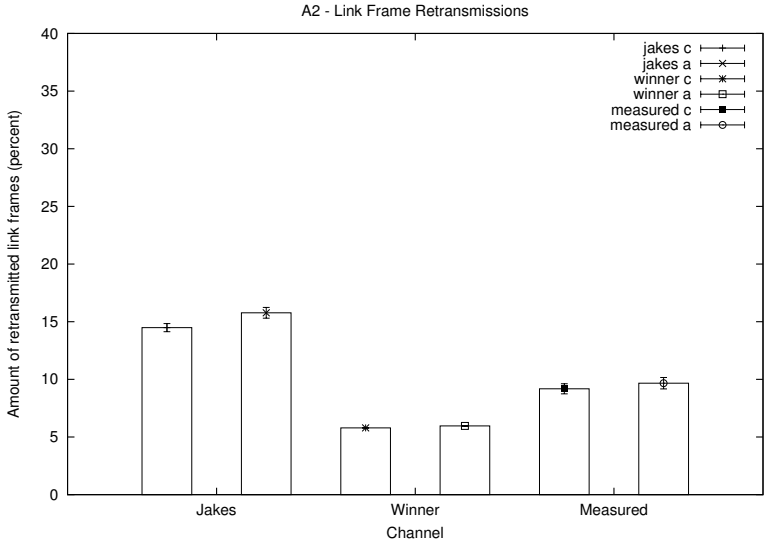
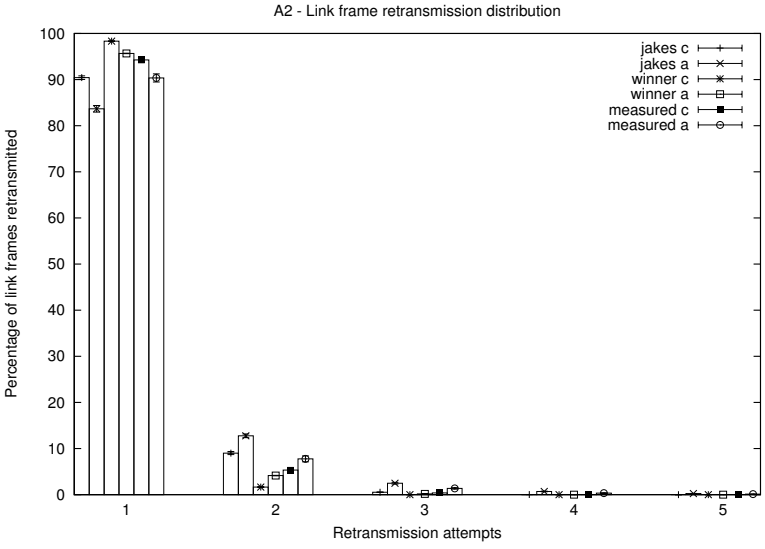
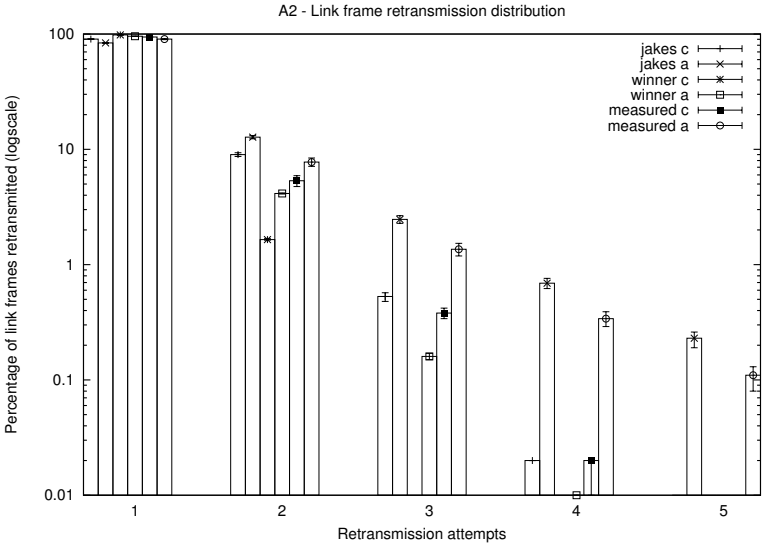


Figure 6.6: Amount of link retransmissions for the Jakes, Winner, and Measured channels with chase combining (labeled "c") and without combining (labeled "a").



(a) Linear scale



(b) Logarithmic scale

Figure 6.7: Link frame retransmission distribution of the Jakes, Winner, and Measured channels with chase combining (labeled "c") and without combining (labeled "a").

retransmissions, but since this number is already low there is little room for improvement. If a combination of two or more frames results in a successful decode, it reduces the number of needed frame retransmissions. From a link capacity point of view, this itself is a marginal improvement. From the TCP point of view, it results in a shorter packet delay. As shown in other experiments with varied link retransmission delay, this has little impact as long as TCP has packets buffered for transmission, and this explains why the symbol combining shows very minor improvement in terms of increased throughput.

6.2.3 Channel prediction

As mentioned in the previous chapter, a core function of the Wireless IP system is the use of adaptive modulation. This is dependent upon a channel prediction to decide which modulation level to use, as discussed in Chapter 5. For the experiments done so far, perfect channel knowledge (perfect prediction) was used. In this section we compare perfect prediction to varying degrees of prediction uncertainty, with different adaptive modulation switch thresholds. These results are partly published earlier in [10]. Further reading on channel estimation and prediction is in [19].

A prediction, by definition, carries a certain amount of uncertainty. The adaptive modulation switch tables used in the evaluations are optimized for a normalized mean squared error of 0 and 0.1 [210]. However, the actual prediction error may not match this exactly. The prediction performance is, among other variables, dependent on the prediction horizon, carrier frequency and the velocity of the terminal. The two first are set in the system design. The terminal on the other hand has a variable velocity (e.g. standing still, walking, driving) which leads to a varying channel prediction performance. To analyze this effect, the channel data includes a varying amount of channel prediction error, ranging from a perfect prediction performance of NMSE 0 to prediction performance of NMSE 0.2 which corresponds to twice the error of the switch table optimization.

To reiterate, there are two aspects to keep separate: First, the

channel prediction error which is a continuous function mainly dependent on the predictor and channel conditions. Second, the *adaptive modulation switch table*, that is pre-computed and optimized for a specific target prediction error (here NMSE 0 and NMSE 0.1). More optimization levels would be possible to use, but results below indicate that there is not much room for improvement.

The experiments combine these two parameters to investigate the impact of the actual prediction error versus the target prediction error of the adaptive modulation switch table.

Experiment results

Fig. 6.8 shows the TCP throughput as a function of the discussed channel prediction errors and modulation switch tables, for the Winner, Measured and Jakes channels. The figure indicates that the system is quite robust against a varying amount of channel prediction error.

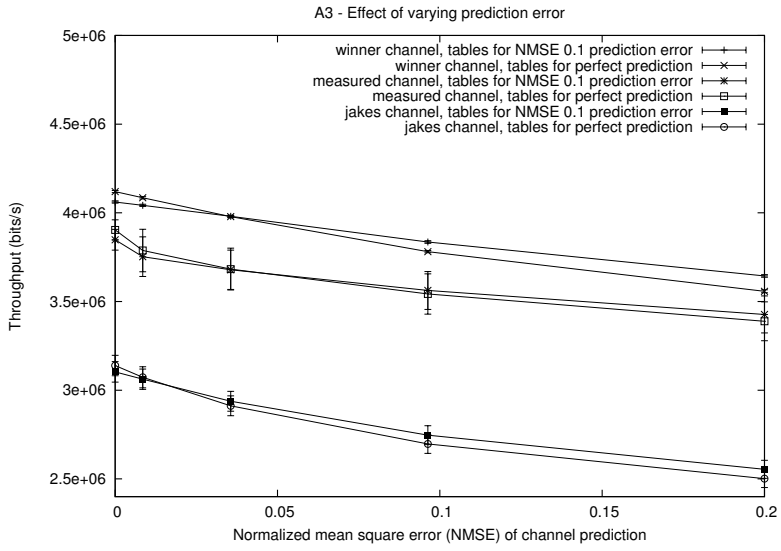


Figure 6.8: TCP throughput as a function of the channel prediction error, for the three channels, with the adaptive modulation switch table optimized for either perfect prediction or a prediction error of NMSE 0.1.

As expected, when the modulation switch levels are optimized for NMSE 0 and the channel prediction error is the same, it results in higher throughput than using switch levels optimized for NMSE 0.1. Conversely, switch levels optimized for NMSE 0.1 results in higher throughput than switch levels optimized for NMSE 0 when the channel prediction error is around NMSE 0.1. Between channel prediction errors NMSE 0 and NMSE 0.1 there is a cross-over section where they perform the same.

Intuitively, the modulation switch levels should be optimized for the currently estimated channel prediction error. This would require a matrix with a switching table for all prediction errors. However, the results indicate that the difference between switch levels optimized for NMSE 0 and NMSE 0.1 is fairly small. The prediction error itself has a larger impact than the switch levels. The channel condition (manifested through the three different channel types) also has a much larger impact, as well as the channel allocation strategy shown in the next section.

The reason that the switch table optimization does not have a greater impact is because the switch intervals overlap for the most part, and because of the of the modulation index granularity. This is illustrated in Figure 6.9. The switch levels optimized for perfect prediction (the leftmost ranges) switch at a slightly lower SNR compared to the levels optimized for a NMSE 0.1 prediction error. This means that when the channel is known to have a prediction error, the switch levels are more conservative, i.e., a higher predicted SNR is required before choosing a higher modulation order. Better performance is achieved by using a lower modulation when the predicted channel is at the border of a switch threshold. A channel prediction with an error could otherwise cause a higher modulation level to be used, leading to an increased risk of symbol errors. However, a difference in performance between the NMSE 0 and 0.1 switch thresholds only occurs when a channel prediction is in the non-overlapping region. Since this non-overlapping range is small compared to the overlapping parts, so is the performance difference between the two tables. This is also impacted by the granularity, or the number of switch levels. By increasing the number of intervals, for example by adding

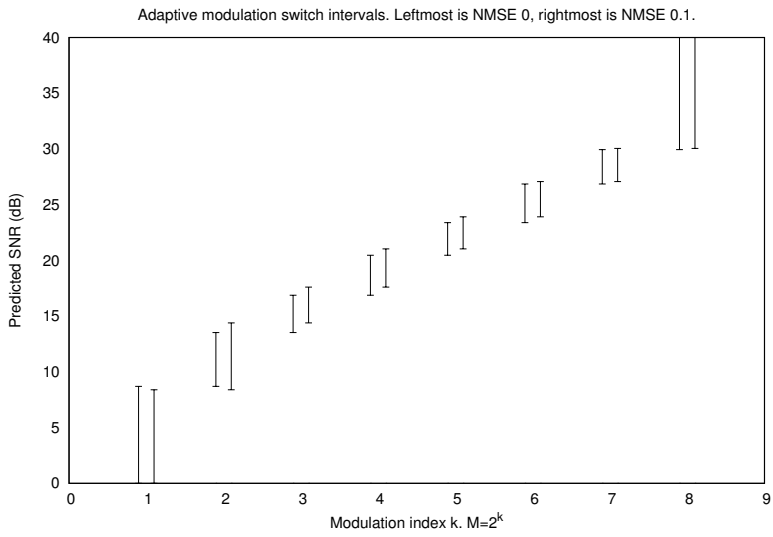


Figure 6.9: Adaptive modulation switch intervals from [210], optimized for NMSE 0 (leftmost for each modulation index) and NMSE 0.1 (rightmost in each modulation index) for a mean channel SNR of 16 dB.

coding, there would be more non-overlapping ranges to provide more opportunity to choose a more exact modulation/coding scheme.

6.2.4 Channel allocation

The achieved capacity is dependent on the channel allocation. During a time interval, the transmission spectrum experiences varying signal strength[3] over different frequencies. Thus, a user that is allocated bins with a high signal-to-noise ratio will, on average, achieve a higher capacity as compared to a user that is allocated a static channel. The allocation strategy can also allocate a varying number of bins in each time instance, in the Wireless IP system between 0-25 bins in a 5 MHz band. In a time interval, all bins can be allocated to different users according to some scheduling strategy. For example, with frequency-selective fading, one user may experience good channel conditions in

[3]Assuming frequency selective fading.

a particular bin/timeslot where another user may experience poor channel conditions. The scheduler can then choose to transmit to the user that can make the best use of that particular bin/timeslot.

The area of channel allocation and scheduling is a research topic in itself and a complete coverage of this topic is outside the scope of this dissertation. Therefore, two very simple allocation schemes are used. The simplest approach is to do no scheduling at all, and transmit at a certain bin index constantly. This would be similar to a FDMA scheme where different frequencies are allocated to different users. This is termed "static allocation" and has been used in the experiments so far.

The optimal approach (at least from the individual user perspective, but not necessarily from a system perspective) would be to transmit in the bin/timeslots with the best channel condition. So if n bins are used for transmission, these will be the n bins with the best SNR values among the 25 available. This is termed "scheduled allocation" in the experiments. As a side note, if the full system capacity is used for a single user there would be no difference between static and scheduled allocation.

These results are partly published earlier in [9, 10].

Experiment results

Figure 6.10 shows the impact of channel allocation for a user that has been allocated ten bins per timeslot. First the bins are chosen according to static allocation, and then the bins are chosen according to scheduled allocation. As shown in the figure, the channel allocation causes a significant difference in achieved throughput.

To further understand the performance differences, we investigate how the utilized modulation levels vary for the static and scheduled allocation schemes. Figure 6.11 shows the normalized amount of link frames that has used a specific modulation level (from BPSK to 256-QAM for the Jakes channel).

For both allocation strategies, we see a spike at 2 bits per symbol (corresponding to 4-QAM). This gives an indication of the mean channel SNR, and if there was a higher mean SNR, the spike would

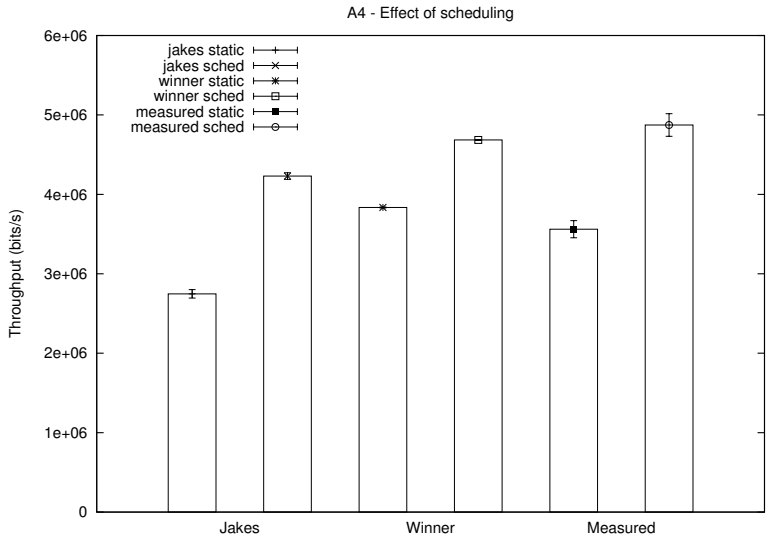


Figure 6.10: TCP throughput of the Jakes, Winner, and Measured channels depending on channel allocation scheme. "sched" indicates scheduled allocation and "static" indicates static allocation.

shift to the right, i.e., higher modulation levels would be chosen more frequently.

Comparing the static allocation to the scheduled allocation, we see the cause of the capacity difference that was seen in Fig 6.10. For static allocation in Fig 6.11, about 20% of the frames use 1 bit/symbol modulation and about 40% of the frames use 2 bits/symbol modulation. For scheduled allocation, these modulation levels are chosen for about 8% and 30% of the frames respectively, i.e., a markedly lower amount than for static allocation. The remaining frames have used higher modulation, leading to higher capacity and higher throughput. This is in line with what can be expected, as the scheduled allocation purposely chooses bins with the highest SNR, and therefore achieves a higher capacity.

To further correlate the experimental results with theoretical results we can compare the spectral efficiency. For example, the Winner channel reaches around 3.5 Mbit/s for static allocation, and 5 Mbit/s for scheduled allocation. Scheduled allocation has a lower

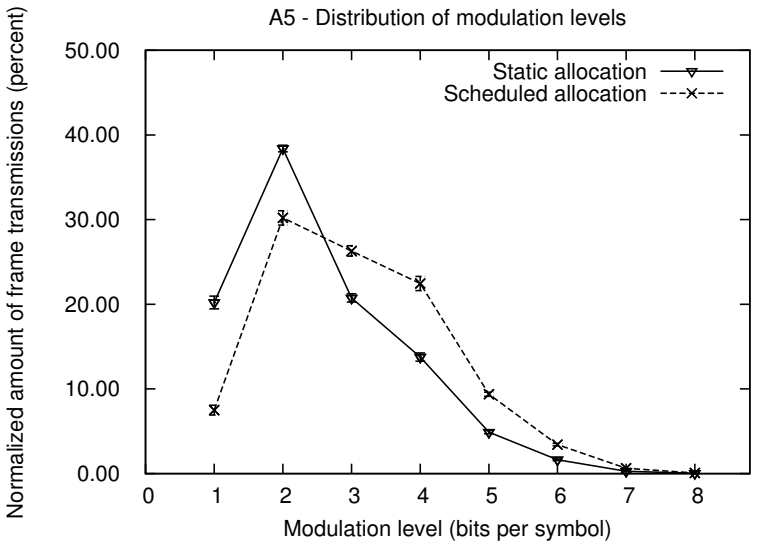


Figure 6.11: Modulation level utilization for static vs. scheduled channel allocation, for the Jakes channel with NMSE 0.1 channel prediction error and modulation switch levels optimized for NMSE 0.1.

variation of channel quality of the bins since the bins with the lowest SNR are avoided, which is reflected in the higher throughput. With 10 of 25 available time slots being used, this corresponds to $\frac{10}{25} * 5 = 2$ MHz. The spectral efficiency is thus in the interval $\frac{3.5*10^6 \ bits/s}{2*10^6 Hz} = 1.75$ bits/s/Hz to $\frac{5*10^6 \ bits/s}{2*10^6 Hz} = 2.5$ bits/s/Hz. This is in line with the theoretical and simulated spectral efficiency in [235] for one receiver antenna and one user, at 16 dB SNR.

6.2.5 Baseline parameter summary

This section investigated the performance impact of fixed versus adaptive modulation, link frame retransmissions, channel prediction errors, adaptive modulation switch level optimization, symbol combining and channel allocation. Based on the findings in the analyzed experiments, the continued experiments will use the following parameter settings as their baseline;

- Unlimited link frame retransmissions using symbol combining are used.
- Adaptive modulation with switch levels optimized for a prediction error corresponding to the actual channel prediction error of NMSE 0.1.
- Channel prediction error realistic for vehicular speeds and prediction horizon of 2 ms (NMSE 0.1).
- Scheduled channel allocation.

6.3 Link and network parameter experiments

After establishing the set of base line parameters, we investigate the impact of secondary or indirect parameters. These are the link retransmission delay, delay in the wired network, packet loss and buffer sizing.

6.3.1 Link retransmission delay

The link retransmission delay is the delay after a first transmission has been received incorrectly, until the retransmission is received. In the Wireless IP system, the design target for the link retransmission delay is 2 ms. This corresponds to 3 timeslots. During the first timeslot, the receiver signals to the sender and requests a retransmission. During the second timeslot the retransmission is scheduled, and if possible, transmitted during the third timeslot. Retransmissions receive priority over non-retransmissions.

The retransmission delay stems partly from the system design, i.e., how many timeslots are needed for signaling, and partly from scheduling, i.e. the designated timeslot may not be the available due to channel conditions. Also, designing for longer retransmission delay can give more time to do for example a more efficient scheduling, or

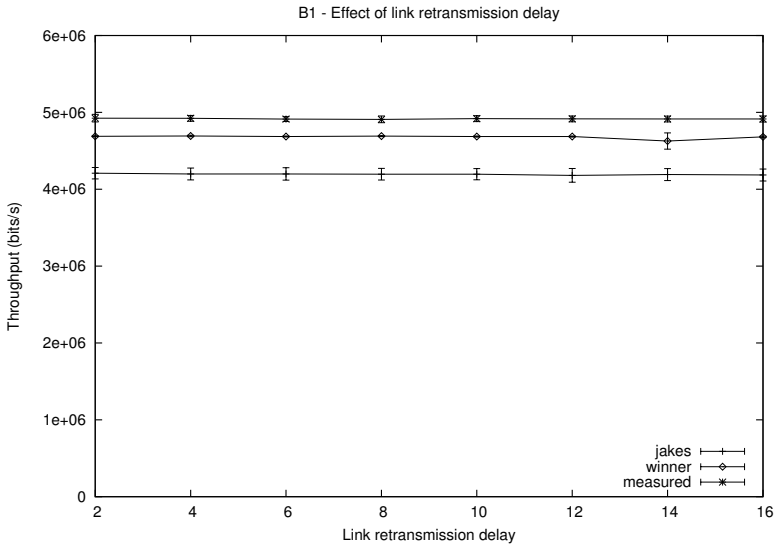


Figure 6.12: Link retransmission delay impact on TCP throughput for the Jakes, Winner, and Measured channels.

use simpler/slower/cheaper circuitry. At the same time, long retransmission delays could cause negative interaction with for example the transport layer [164]. It is therefore interesting to study the impact of various retransmission delays and their effect on the throughput on the transport layer. To this end, a number of experiments were performed with varying link retransmission delay (2-16 ms) while measuring the resulting TCP throughput.

Experiment results

Fig. 6.12 shows the TCP throughput for the three channels, as a function of the link retransmission delay. As seen in the figure, the throughput remains unaffected from the retransmission delay. The reason for this is the following. First, when there are packets buffered for transmission, the channel capacity will be utilized even if some frames are awaiting retransmission. If a frame retransmission takes 2 or 16 ms is therefore a matter of buffering to keep the channel utilized.

Second, the distribution of frame errors is such that almost all packets will have retransmitted frames. A 1500 byte packet uses about 30 link frames (at 16 dB mean SNR). As the frame error rate in this system is between 10-30 percent (depending on the channel), the probability of at least one frame retransmission per packet is high. The implication of this is that most packets would get an additional transmission delay, corresponding to the frame retransmission delay, or multiples thereof.

A third aspect is that of the relation between delays. Given 10 bins per time slot, 30 frames are delivered in 3 time slots. Retransmitted frames would therefore be transmitted after all frames had been delivered, possibly with an idle period if no other packets awaited transmission. If fewer bins per time slot are allocated, for example one bin per time slot, the total packet transmission time would be stretched out. Link frame retransmissions would occur mainly within the transmission time, instead of afterwards, leading to even less impact of the retransmission delay.

This is illustrated with the following example in Fig. 6.13: Consider a retransmission delay of 16 ms and a packet of 1500 bytes split into 30 link frames. The first frame is corrupted, and requires one retransmission which is successful. The other frames are delivered correctly. For the case of "high capacity" transmission of 10 frames per bin, this would increase the minimum transmission time from 2 ms to 17 ms. For the "low capacity" transmission of 1 frame per bin, the minimum transmission time would be 30 $frames{*}0.667\ ms/frame = 20\ ms$. The corrupt frame would therefore be retransmitted within the packet transmission time, and the minimum transmission time would increase by one time slot to 20.667 ms.

A fourth aspect on the frame retransmission delay impact is the relation to the total end-to-end delay. If the end-to-end delay is short, on a scale of tens of milliseconds, the delay variation caused by retransmission will have a larger impact than if the total delay is larger.

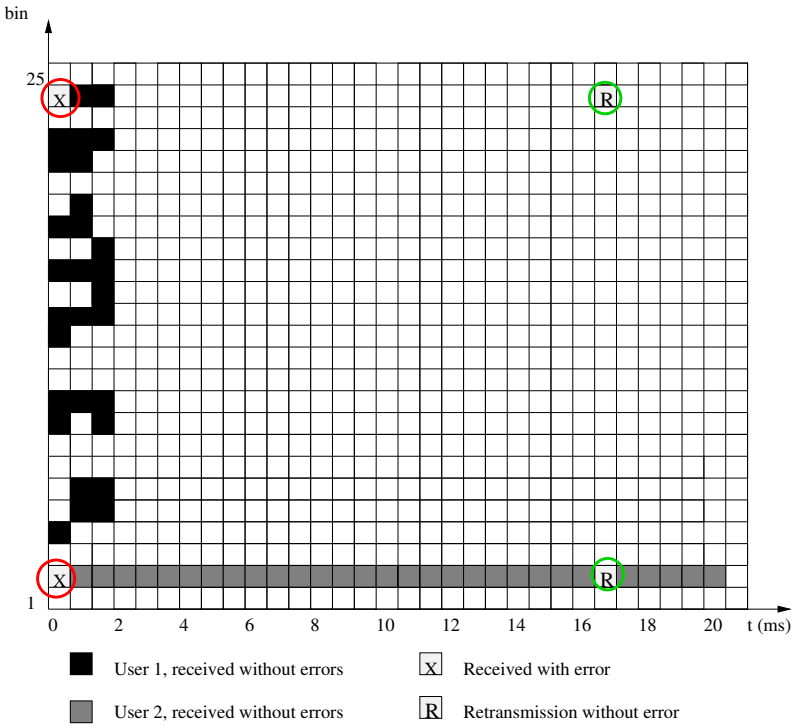


Figure 6.13: Illustration of the effect of link retransmission delay depending on the allocated capacity to a user.

6.3.2 Delay in wired network

In addition to the delay over the wireless link, the connecting wired networks also introduce delay to a varying extent. The amount of delay depends on the distance and propagation speed between the communicating parties, and network queueing. These delays contribute to an increase of the estimated round-trip time, affecting TCP error recovery in the event of packet loss. A delay increase also increases the bandwidth-delay product. The bandwidth-delay product is a measure of how much data can be in flight, but also how much data need to be outstanding for a protocol to operate at maximum capacity. For example, with a bandwidth-delay product of 5 $Mbit/s * 100\ ms = 500\ kbit$, TCP should be able to handle at least this amount of data being unacknowledged by the receiver.

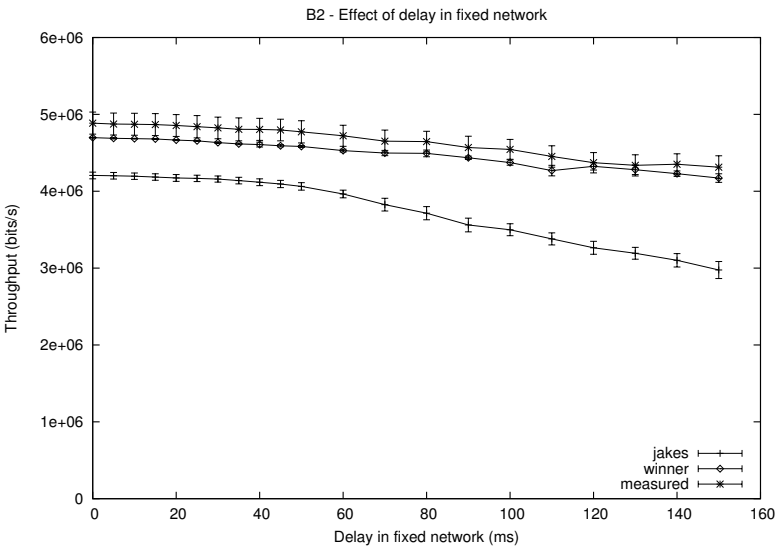


Figure 6.14: Wired network delay impact on TCP throughput for the Jakes, Winner, and Measured channel.

The standard TCP receiver window size (without window scaling) is 64 kB $\approx$ 524 kbit, which is just about enough. However, should the RTT double to 200 ms, this receiver window size would not be sufficient to allow full buffering, leading to an under-utilized network and non-optimal throughput, unless the TCP receiver window is adjusted. To investigate the effect of the wired network delay, a number of experiments were performed with TCP throughput measurement with a varying delay from 0 to 150 ms. These results are partly published earlier in [10].

Experiment results

Fig. 6.14 shows the TCP throughput as a function of the wired network delay. The throughput is largely unaffected until a delay of 50 ms is reached, after which the throughput starts dropping off slightly. The reason for this is due to the transmission time being fixed at 30 seconds. Thus when the delay is increased, the TCP startup phase is prolonged. A longer startup time in the transmission time is then

exhibited as lower throughput.

The steady-state throughput is otherwise not severely affected by increased delay. The reason for this is that as long as packets are buffered in the network and the bottleneck link is fully utilized, throughput, or rather goodput, is decreased by the same magnitude as the retransmissions. For example, 1% percent retransmissions would reduce goodput by the same amount regardless of the delay. If there are idle periods in the transmission, this is more influential on the resulting throughput. Typically this happens at frequent packet losses, and is further discussed in the next section. Buffer sizing also impacts idle periods, and is discussed after the next section.

6.3.3 Loss in wired network

In a live network as opposed to the experiment setup, there will always be packet loss to a certain extent, depending on the general congestion level in the network between the communicating parties, or wireless losses due to radio conditions. These losses are usually recovered with end-to-end retransmissions. In the case of TCP, the sender reduces the transmission rate to mitigate the perceived network congestion. This then leads to a lower throughput. To determine to which extent the throughput degrades, experiments were performed by inducing a varying degree of packet loss and measuring the resulting throughput. With regard to cross-layer interactions, the experiments also vary the link retransmission delay, and it is shown to have a notable impact.

Experiment results

Fig. 6.15 shows the TCP throughput as a function of an increasing amount of packet loss, from 0 to 10%, for a link retransmission delay of 2, 8, and 16 ms, for the Jakes channel. As expected, a general observation is that the throughput degrades quickly in response to the loss. A few percent packet loss results in halved throughput. This points to the importance of keeping packet losses to a minimum wherever possible. While true congestion related packet loss cannot

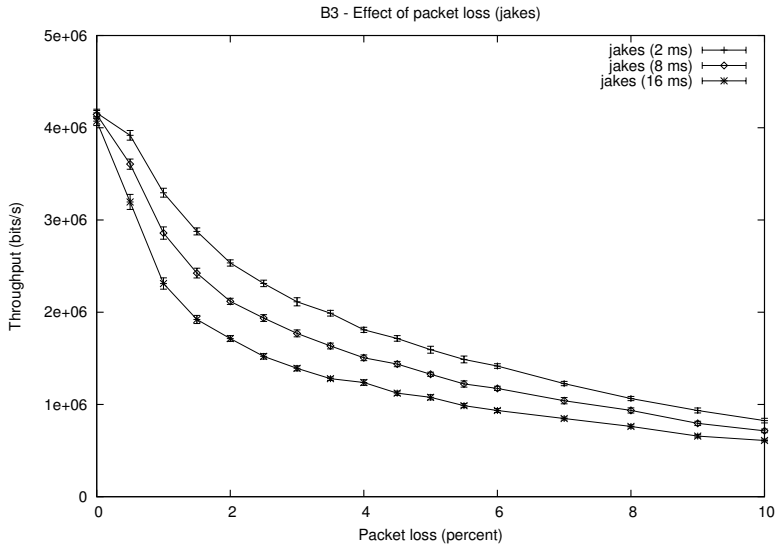


Figure 6.15: The effect of packet loss on TCP throughput for the Jakes channel with a link retransmission delay of 2, 8, and 16 ms.

be avoided, the wireless part of the network path should be configured to provide as reliable delivery as possible.

An interesting observation is the effect of the link retransmission delay in the event of packet loss. In the previous section (6.3.1), throughput remained the same when the link retransmission delay was varied. Already at low amounts of packet loss, a low link retransmission delay is clearly beneficial, as seen in Fig 6.15. The explanation for this is that the increased link retransmission delay increases the round-trip time. Without packet loss this does not impact the throughput (cf. Section 6.3.1). With packet loss, the increased round-trip time causes longer response times for the senders congestion control.

To verify these curves, a comparison of the throughput is made to a widely used analytical model [172, Eq. 32]:

$$B(p) \approx min\left(\frac{W_{max}}{RTT}, \frac{1}{RTT * \sqrt{\frac{2bp}{3}} + T_0\ min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)}\right) \quad (6.2)$$

Loss probability p	$B_{model}(p)$	$B_{jakes}(p)$
0.01	3.31 Mbit/s	3.30 Mbit/s
0.02	2.04 Mbit/s	2.53 Mbit/s
0.03	1.47 Mbit/s	2.11 Mbit/s

Table 6.1: Comparison of analytical and measured TCP throughput of the Jakes channel, 2 ms retransmission delay.

where $B(p)$ is an approximation of the TCP throughput as a function of the number of loss events p, maximum receiver window W_{max} (64096 extracted from experiment packet captures), round-trip-time RTT (26.7 ms as calculated by tcptrace from experiment packet captures, with 2 ms link retransmission delay used), average time of a single timeout T_0 (200 ms, set as `TCP_RTO_MIN` in `include/net/tcp.h` in the Linux kernel source), and b packets acknowledged by an ACK (2 packets with delayed acks).

Table 6.1 shows that the analytically derived throughput and the measured throughput match up fairly well. The measured throughput is higher than the calculated throughput at higher loss rates. A plausible explanation is that the TCP stack is using features that are not captured in the formula, such as fast retransmit/recovery and selective acknowledgements that avoids retransmission timeouts, and therefore achieves better performance.

Finally, Figures 6.16 and 6.17 show the result for the Winner and Measured channel, respectively. They exhibit the same characteristics as discussed for the Jakes channel.

To show the effect of packet loss further, we examine two individual TCP sessions in detail. Figure 6.18 shows a tcptrace time-sequence graph of a TCP transmission without artificial[4] packet loss. The graph shows sent TCP segments, received acknowledgements and the receiver window. Figure 6.19 shows a close up of a part of the transmission without packet loss. As acknowledgements arrive, new packets are sent out in a steady stream. The throughput is about 4 Mbit/s.

[4]There is some packet loss, but it is due to full buffers in the network.

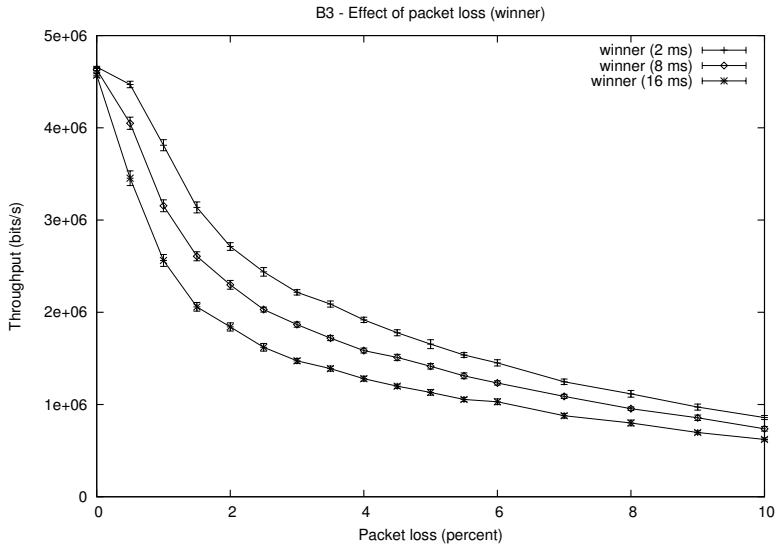


Figure 6.16: The effect of packet loss on TCP throughput for the Winner channel with a link retransmission delay of 2, 16 and 32 ms

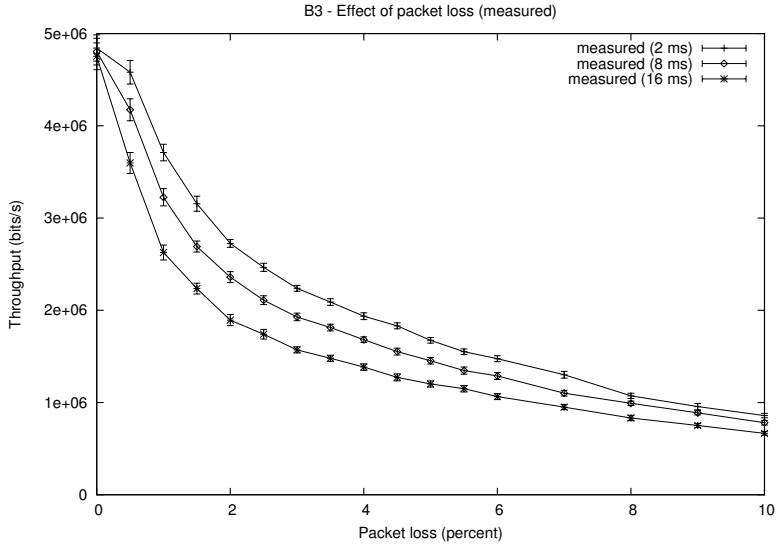


Figure 6.17: The effect of packet loss on TCP throughput for the Measured channel with a link retransmission delay of 2, 16 and 32 ms

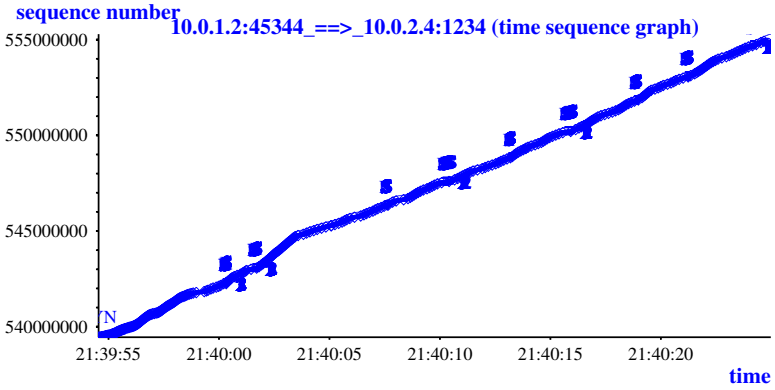


Figure 6.18: Time-Sequence graph of a TCP session without artificial packet loss.

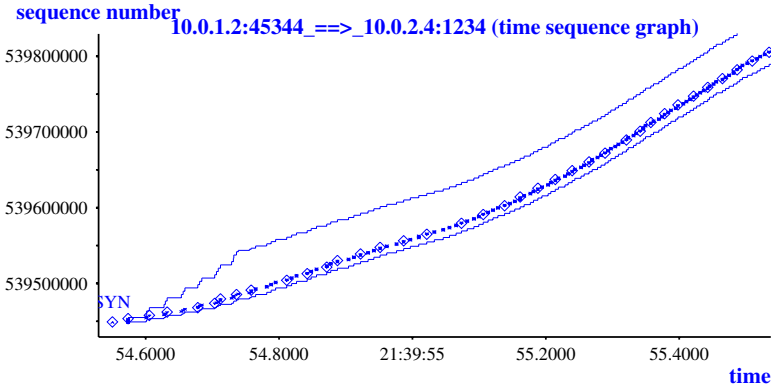


Figure 6.19: Time-Sequence graph of a section of a TCP session without artificial packet loss.

This can be compared to Figures 6.20 (full) and 6.21 (zoom), which show the same transmission but with 5% induced packet loss. Here TCP segments are frequently selectively acknowledged, denoted by the S flag, indicating that a segment has been lost and after which it is being retransmitted, denoted by the R in the graph. The packet loss causes a substantial throughput degradation to about 1.6 Mbit/s.

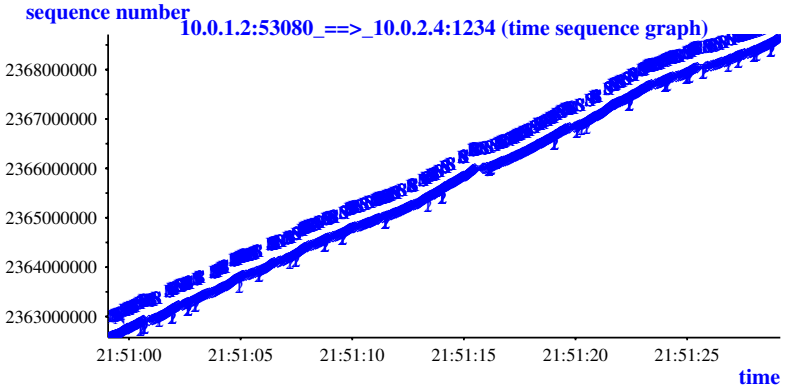


Figure 6.20: Time-Sequence graph of a TCP session with 5% induced packet loss.

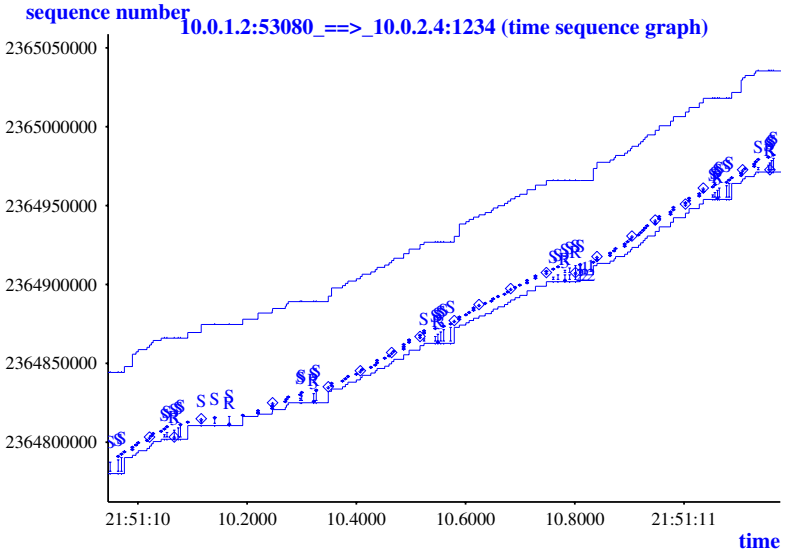


Figure 6.21: Time-Sequence graph of a section of a TCP session with 5% induced packet loss.

6.3.4 Buffer sizing

To be able to fill the capacity of the wireless link, there should always be data available for transmission. To achieve this, a buffer is used to queue packets. A question is then how small or large such a buffer should be. A rule-of-thumb [40, 79] is to use the bandwidth-delay product of the link for the buffer size[5]. This means that there is data buffered to keep the link utilized for the duration of a round-trip time, enough for the TCP congestion control to keep feeding the buffer and avoid under-utilization. The aspects of buffer sizing are extensively discussed in [78].

The problem for a wireless link is that the bandwidth and delay varies depending on the channel condition, so an exact optimal buffer size is impossible to set. To investigate the impact of the buffer sizing for the Wireless IP system, experiments have been performed with varying buffer sizes and delays while measuring the resulting throughput of bulk data transfers.

Experiment results

The experiments were performed with 30 second TCP transfers, for each of the three channels, and replicated 30 times. The buffer size was varied between 1 to 19 packets in steps of one packet, and the link retransmission delay was varied between 2 to 16 ms in steps of 2 ms. The wired network delay was fixed at 10 ms. A reasonable buffer size B according to the rule-of-thumb, where the capacity is 4 Mbit/s and the RTT is 20 ms, would be

$$B = C * RTT = 4 * 10^6 \; bits/s * 20 \; ms = 8 * 10^4 \; bits = 10^4 \; bytes$$

$$10^4 \; bytes \Rightarrow 10^4 \; bytes/1500 \; bytes/packet \approx 7 \; packets$$

[5]For high-speed core routers with tens of thousands of flows this would require very large buffers, so the rule-of-thumb is not suitable for such scenarios; instead it can be divided by the square root of the number of flows as shown in [18].

Since an RTT of 20 ms only accounts for the delay in the wired network, the calculated buffer size should be regarded as a minimum. Should the RTT double, so should the buffer size.

Figure 6.22 shows a graph of the TCP throughput as a function of the buffer size (packet queue length) and the retransmission delay for the wireless link, for the Jakes channel. It shows a clear performance degradation when the buffer size is constrained. The link retransmission delay also has an impact, mainly for the region where the buffer size is 3-7 packets. The reason is that a higher link retransmission delay increases the round-trip time, thus increasing the needed buffer size to keep a continuous transmission. When larger buffer sizes are used (i.e., 19 packets), the effect of the link retransmission delay is of lesser importance. For completeness and to show coherence, Fig 6.23 and 6.24 show the corresponding results for the Winner and Measured channel models. They show the same characteristics as the Jakes channel, although the absolute throughput differs slightly.

To get a better understanding of the impact of the buffer size, Fig 6.25 shows a two-dimensional view for the 2 ms link retransmission delay, for the three channels. As expected, a buffer size below 7 packets severely decreases the throughput, while the throughput flattens out at a buffer size of about 7-10 packets (variation is due to channel capacity and varying amount of link retransmissions that affect the delay).

The conclusion of these experiments is that the buffer size plays an important role in the achieved TCP throughput, and should be roughly equal to $C * RTT$ (as shown by the previously cited research) for this scenario with a single flow. The challenge is that the channel capacity of the wireless link fluctuates, and therefore the buffer requirements. The buffer size should therefore have a margin to accommodate these variations. If there are concurrent flows over the link, these should also be taken into consideration. At the same time buffer size should not be larger than necessary, as this introduces additional queueing delay and longer time for recovery of packet losses [17]. Besides the buffer size, the link retransmission delay also impacts the throughput in the event of packet losses. The link retransmission delay should therefore be kept as low as possible.

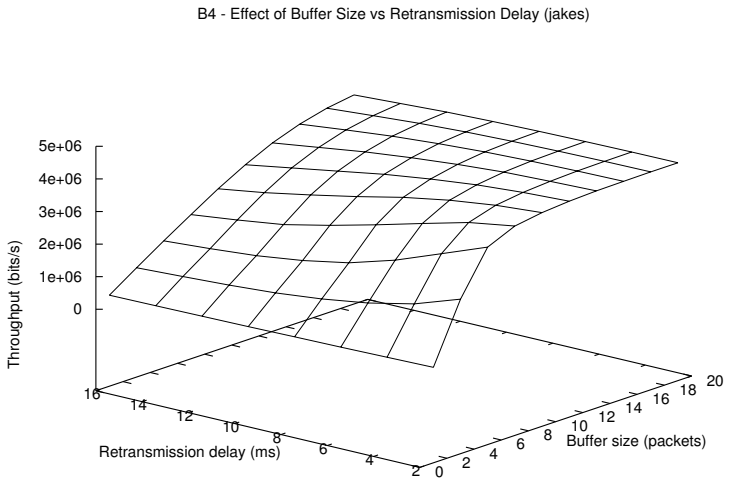


Figure 6.22: The impact of buffer size and retransmission delay on TCP throughput for the Jakes channel.

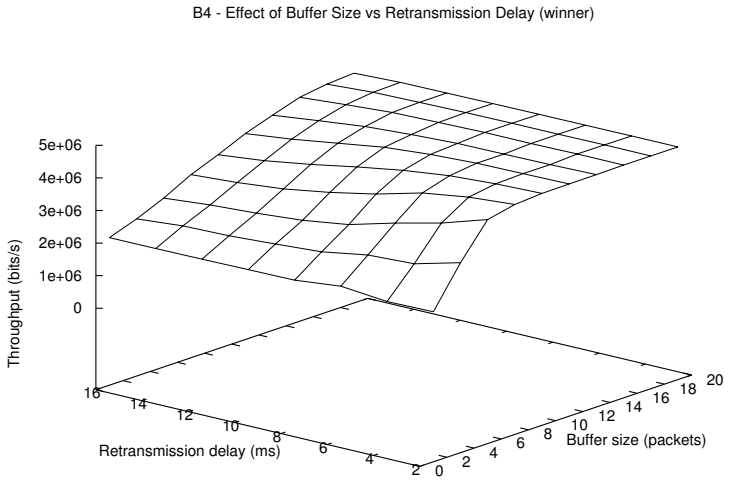


Figure 6.23: The impact of buffer size and retransmission delay on TCP throughput for the Winner channel.

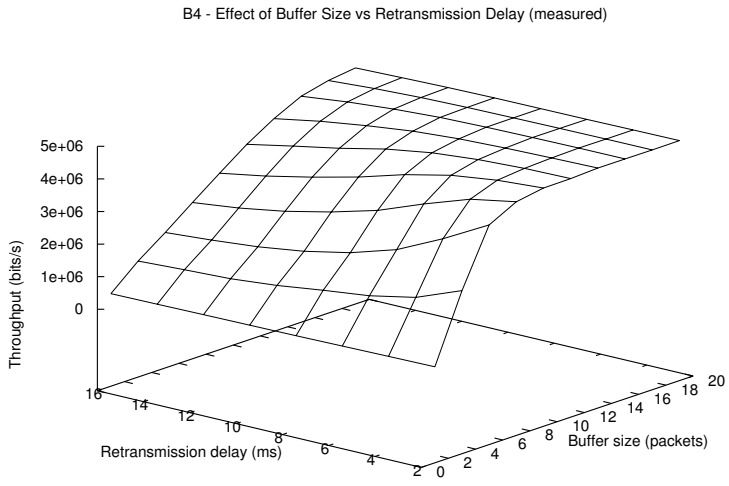


Figure 6.24: The impact of buffer size and retransmission delay on TCP throughput for the Measured channel.

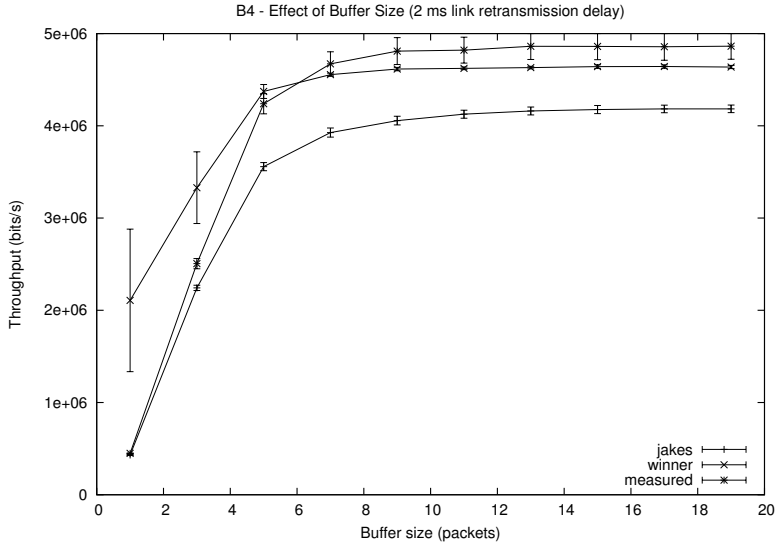


Figure 6.25: A closer inspection on the buffer size impact on TCP throughput, for 2 ms link layer retransmission delay for all channels.

6.4 Application parameter experiments

Different applications have different requirements on the service provided by the transport and network layer [204, C2.6]. A common trade-off is that of reliability and delay. Increased reliability can be achieved with retransmissions or by adding forward error correction, at the cost of adding delay.

In this section we investigate two different classes of applications. The first application, file transfer, requires a reliable transport service and tolerates a fluctuating throughput. The second application, VoIP (Voice over IP), may accept an unreliable transport service with a limited amount of packet loss, but has demands on a minimum throughput, packet delay and delay variation. Experiments on WIPEMU and VoIP performance are partly published earlier in [11]. These traffic classes also resemble those that are used for evaluating IMT-Advanced [109, A.2], where a "full buffer model" and "VoIP traffic model" is used.

6.4.1 File transfer size experiments

The goal of a file transfer is to reliably transfer an amount of data at maximum throughput, although a varying throughput can be acceptable. The application is still functional with a low throughput, although the transfer takes a longer time to complete. The experiments in previous sections used a 30 second data transfer period to allow TCP to reach a steady-state with full link utilization. As shown in Section 6.3.1 and 6.3.2, the link retransmission delay and the delay in the wired network had no significant effect on the TCP throughput for this scenario. However, the investigation of the wired delay showed that long round-trip times contribute to a longer time before the channel is fully utilized, which leads to a throughput degradation. The amount of degradation depends on the relation between the delay and the flow length[6].

[6]There is research [13, 62] investigating the effect of increasing the initial congestion window which would allow for more data to be sent initially.

Short flows are commonly found when web browsing, where a web page may consist of a few kbyte of HTML data, further linking images and scripts of hundreds of kilobytes to megabyte sizes. For example, the www.google.com search page is 9424 bytes, the www.cnn.com front page is 85488 bytes and the www.dagensnyheter.se front page is 369876 bytes (as of 15 March 2011). These figures only include the base HTML page and are subject to change, but give a hint of the sizes involved. In [181], researchers at Google report that the average web page is 320 kbyte including HTTP headers (as of May 2010) from a sample of 4.2 billion pages. The 10th to the 90th percentile ranges from 22 kbyte to 663 kbyte.

To show the effect of flow size and delay on TCP performance in the context of the Wireless IP system, a number of experiments have been performed with different data sizes in combination with varying link retransmission delay and wired network delay, described in the next section.

The performance metric used to present the results is throughput, although a more comprehensible metric for short flows is completion time [61]. For example when web browsing, the page load time is easier to relate to than a specific throughput. The throughput of short flows, as shown in the next section, could also be far from the throughput of longer flows. On the other hand, for longer flows or downloading of unspecified data sizes, the throughput metric is more sensible. However, as the completion time is only comparable between the same data sizes, throughput is used in the experiments to normalize the results to a comparable metric between data sizes. As throughput and completion time are inversely related, the completion time is still implicitly available.

Experiment results

Figure 6.26 shows the TCP throughput depending on the data size being transferred (flow size), and the delay in the wired network, for the Jakes channel with 2 ms retransmission delay. The data sizes used are 1, 10, 100, 1000 and 10000 kbyte. For smaller data sizes below 1000 kbyte, the graph shows a sharp drop off in the throughput,

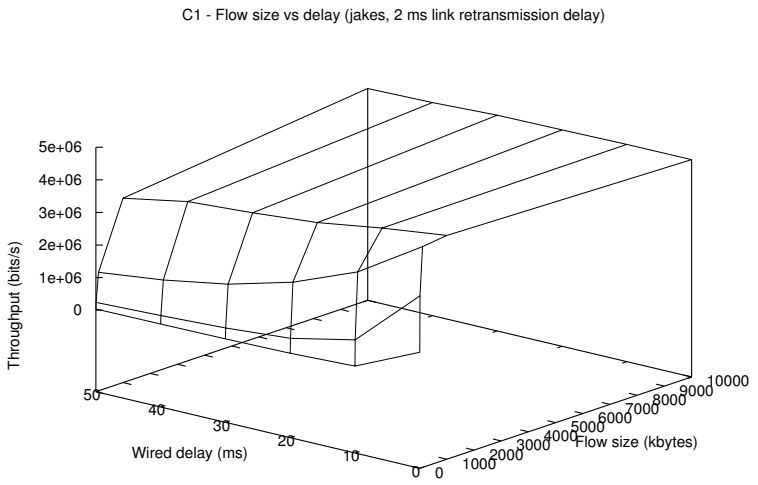


Figure 6.26: Throughput as a function of TCP flow size and delay in wired network for the Jakes channel with 2 ms link retransmission delay.

which also falls off as the wired delay (and therefore the round-trip time) increases.

To illustrate the delay impact, consider the smallest data size of 1 kbyte. This amount of data can be transmitted with only one TCP segment, assuming a standard segment size of 1400-1500 bytes, see the trace in Figure 6.27. For the transmission to complete, the session consists of a SYN packet from the sender (0.5 RTT), replied with a SYN+ACK by the receiver (+0.5 RTT), and responded by the sender with another ACK (+0.5 RTT) that completes the TCP handshake. The sender then continues to transmit the data directly after the ACK, and since there is no more data the sender sends a FIN to close the connection. The receiver gets the data, sends an acknowledgement of the data (+0.5 RTT) and an acknowledgement and FIN to close the connection. The sender acknowledges the closing of the connection (+0.5 RTT) and the session is finished. It therefore takes on the order of 2.5 RTT to transmit a single segment.

Returning to Figure 6.26, a data size of 10000 kbyte reaches about

```
snd.54214 > rcv.1234: S 1546203355:1546203355(0) win 5840
                      <mss 1460,sackOK,timestamp 543576618 0,nop,wscale 5>
rcv.1234 > snd.54214: S 1262605521:1262605521(0) ack 1546203356 win 5792
                      <mss 1460,sackOK,timestamp 543564472 543576618,nop,wscale 5>
snd.54214 > rcv.1234: . ack 1 win 183 <nop,nop,timestamp 543576638 543564472>
snd.54214 > rcv.1234: P 1:1025(1024) ack 1 win 183 <nop,nop,timestamp 543576638 543564472>
snd.54214 > rcv.1234: F 1025:1025(0) ack 1 win 183 <nop,nop,timestamp 543576638 543564472>
rcv.1234 > snd.54214: . ack 1025 win 245 <nop,nop,timestamp 543564493 543576638>
rcv.1234 > snd.54214: F 1:1(0) ack 1026 win 245 <nop,nop,timestamp 543564493 543576638>
snd.54214 > rcv.1234: . ack 2 win 183 <nop,nop,timestamp 543576660 543564493>
```

Figure 6.27: tcpdump output on the sender side of a complete TCP session transmitting 1 kbyte of data.

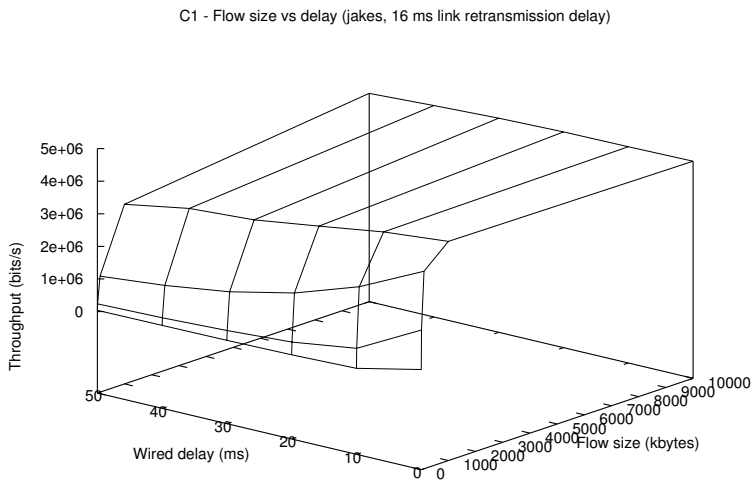


Figure 6.28: Throughput as a function of TCP flow size and delay in wired network for the Jakes channel with 16 ms retransmission delay.

4.1 Mbit/s throughput, consistent with the throughput reached in previous experiments. Since this flow length fully utilizes the channel, the wired network delay has negligible impact compared to the 1 kbyte flow.

Apart from the wired delay, the link retransmission delay is also affecting the throughput of short flows. Figure 6.28 shows the same experiments as figure 6.26, but with a 16 ms link retransmission delay.

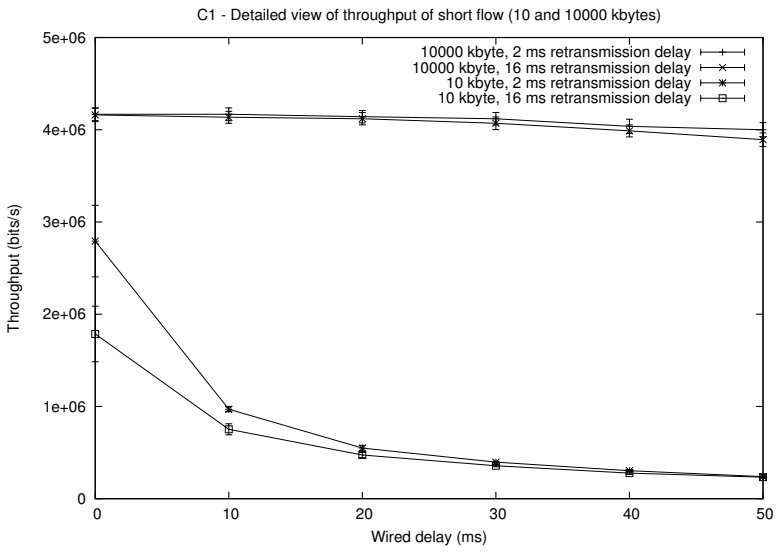


Figure 6.29: Throughput as a function of wired delay and link retransmission delay for flows of 10 and 10000 kbytes for the Jakes channel.

The shorter flows again experience lower throughput compared to the longer flows, and the throughput of shorter flows is also decreased as the wired delay increases. The graph shows that the throughput is generally lower for comparable wired delays. The reason for this is that the link retransmission delay adds a delay component over the wireless part of the network. In effect, this can be considered an offset of the wired delay.

To more clearly show the relation between delay and flow size, figure 6.29 shows a two-dimensional representation of selected data from figures 6.26 and 6.28. In the figure, the 10000 kbyte flow is marginally affected by the link retransmission delay and wired delay. The throughput of the 10 kbyte flow is, on the other hand, highly sensitive to the delay of the wired network and to the link retransmission delay. As the wired delay increases, the effect of the link retransmission delay decreases. For completeness, the results for the Winner and Measured channel are shown in Figures 6.30 – 6.33, exhibiting the same properties as the Jakes channel.

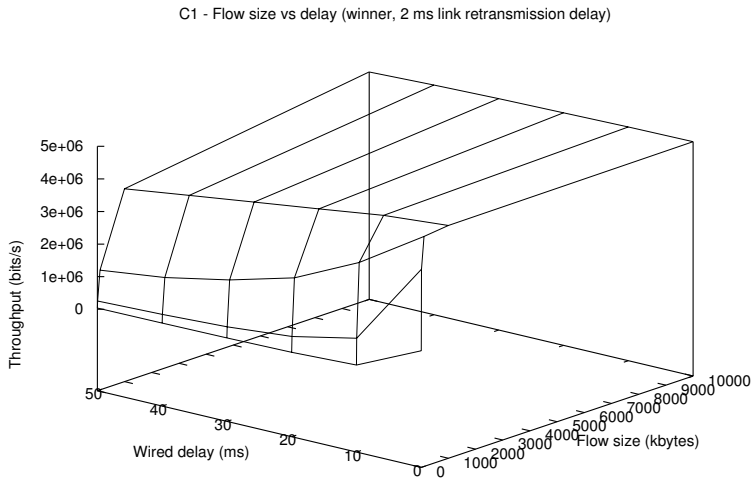


Figure 6.30: Throughput as a function of TCP flow size and delay in wired network for the Winner channel with 2 ms retransmission delay.

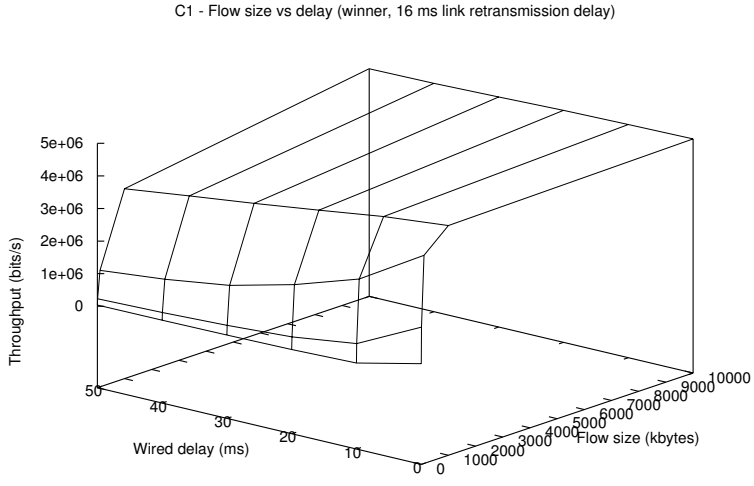


Figure 6.31: Throughput as a function of TCP flow size and delay in wired network for the Winner channel with 16 ms retransmission delay.

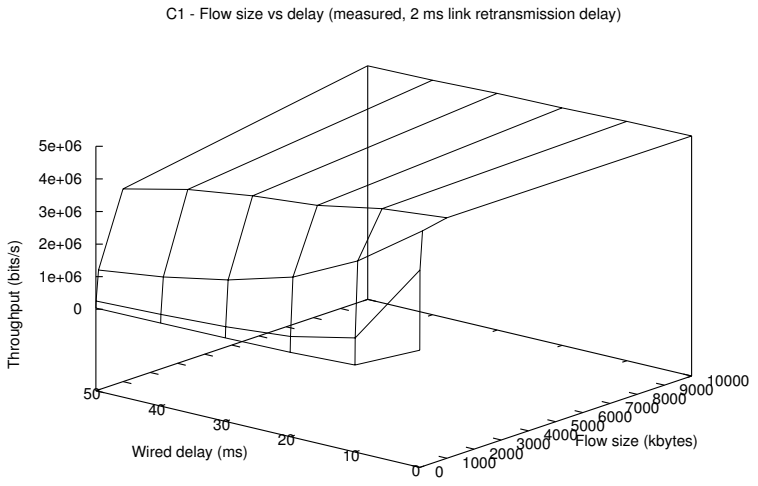


Figure 6.32: Throughput as a function of TCP flow size and delay in wired network for the Measured channel with 2 ms retransmission delay.

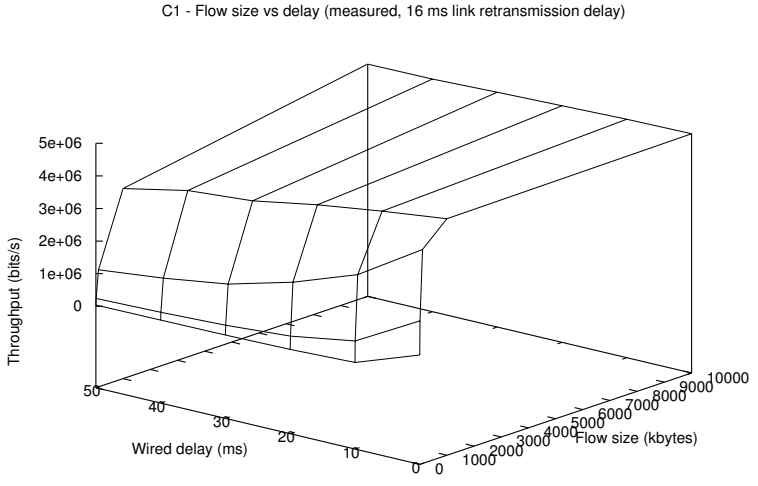


Figure 6.33: Throughput as a function of TCP flow size and delay in wired network for the Measured channel with 16 ms retransmission delay.

6.4.2 VoIP experiments

The goal of an interactive real-time VoIP (Voice over IP) application is to replace the service of a circuit switched telephony service [60]. The circuit switch originally provided a direct electrical connection between two parties, but is since many years replaced by packet switching technologies such as ATM and IP.

Sending voice with packet switching involves a number of steps. First a short sample, 20 ms being common [194], of voice data is recorded and digitized. This is packetized and sent to the receiver where it is unpacketized and the analog signal is played back. When the next packet arrives, it is unpacketized and that sample is played back, and so on.

To provide a continuous playback, the receiver must utilize some amount of buffering. After a sample is played back the following sample should be available and ready for playback. If not, then there will be an interruption of the conversation. In the bulk transfer case, this is handled with buffering. Packets are buffered until all packets have arrived in order, after which it is delivered to the application. For real-time applications, the buffering equals to a conversation delay [132, Ch. 7.3].

If a packet with a voice sample has not arrived, but packets following that, it could be because of packet reordering (solved with more buffering at the expense of added playout delay). A more probable cause of non-arrival is congestion or wireless related packet loss. For a reliable transport protocol, such as TCP, packet loss is handled by doing retransmission of the lost data. This incurs an additional delay, which, depending on the round trip time and delay bounds, can be unfeasible for a real-time application. If the sample time is short, the playback can be silenced, interpolated or have comfort noise generated for that sample (known as packet loss concealment [224]), but the overall quality of the communication is still acceptable. Various coding schemes can also be used to recover lost packets. However, at some amount of packet loss there will be unacceptable quality degradation.

There are perceptual models and metrics that attempt to cap-

ture the experience of the user, as a result of the different delays and losses. Examples of these include Mean Opinion Score (MOS) [111], Perceptual Evaluation of Speech Quality (PESQ) [113], and the E-Model [112]. In this dissertation the packet loss and delay are examined without correlating to speech quality, other than recognizing the recommendations of keeping a low delay and limiting the amount of packet loss. In [224] (fig. 2), delays up to 150 ms are considered "very satisfactory", with a steep slope at 175 ms after which the quality is considered "satisfactory", "some users dissatisfied", "many users dissatisfied" and "exceptional limiting case", on a declining scale up to 500 ms. The explanation for this is that in a normal conversation, there is about 200 ms between speaking turns[7]. Additional delay can be interpreted as hesitation, cause over-talking, or break-in problems. Packet loss on the other hand more directly impacts the speech quality. A loss of a packet can cause pops, clicks or noise depending on the packet loss concealment method employed by the application. As shown in [224, Table 2], a standard G.711 (pulse code modulation or PCM) voice codec with packet loss concealment can tolerate about 3% packet loss while still providing "satisfactory" user experience. These numbers are also consistent with the 3GPP delay budget within the access stratum [1] that lists $< 150\ ms$ as "preferred" one way delay for a conversational voice application, with less than 3% frame loss.

To summarize, there are three important performance parameters for a VoIP service: the throughput requirement to support the voice stream, amount of buffering/delay to recover delayed packets, and the amount of tolerable packet loss. Jitter (delay variation) is also a parameter to consider, but is here assumed to be included in the total delay parameter. In this application, a packet outside the upper delay bound is treated as a packet loss, even if it eventually arrives. The playout delay and packet loss therefore interact. A low playout delay can cause lost samples that a higher playout delay would have included [132, Ch. 7.3].

The question is then how a VoIP application would perform in

[7]It should be noted that time intervals between turn-taking and pacing in a conversation can vary depending on culture [217].

the Wireless IP system, with regards to throughput, delay and loss. For our experiments we assume a VoIP system based on G.711 RTP streaming [194]. This means 20 ms long voice segments represented by packets that include voice samples at 8 kHz with a resolution of 1 byte per sample. This results in 50 packets per second and 160 bytes payload excluding protocol headers, 172 bytes including RTP headers and 200 bytes including UDP/IP headers. G.711 is a simple encoding, and there are more bandwidth-efficient voice codecs in use; for example GSM-EFR [66] that operates at about 12.2 kbit/s, compared to 64 kbit/s for G.711. However, since the experiments focus on packet loss and delays, they are not dependent on the actual codec in use even though different codecs may have different loss and delay requirements. Using G.711 should provide an upper bound for voice-oriented codecs.

VoIP packet loss

The retransmission of link frames received with symbol errors is a source of delay and delay variations. If there is an upper bound on the amount of retransmissions RT_{max} with a retransmission delay of RT_{delay}, there will also be an upper bound on the delay D: $D_{max} = RT_{max} * RT_{delay}$. The trade-off is that packets that require more link retransmissions than the upper limit are dropped and are counted as packet loss.

Figure 6.34 explores this trade-off. The packet loss is plotted as a function of the maximum allowed link frame retransmissions. When the retransmissions are limited, there is a considerable amount of lost packets. The packet loss falls off quickly as retransmissions are performed, and after three retransmissions almost all packets have arrived[8]. The next question is therefore how the delay is affected by the frame retransmissions. Do the retransmissions cause unacceptable delays?

[8]The actual numbers varies depending on the modulation and coding scheme.

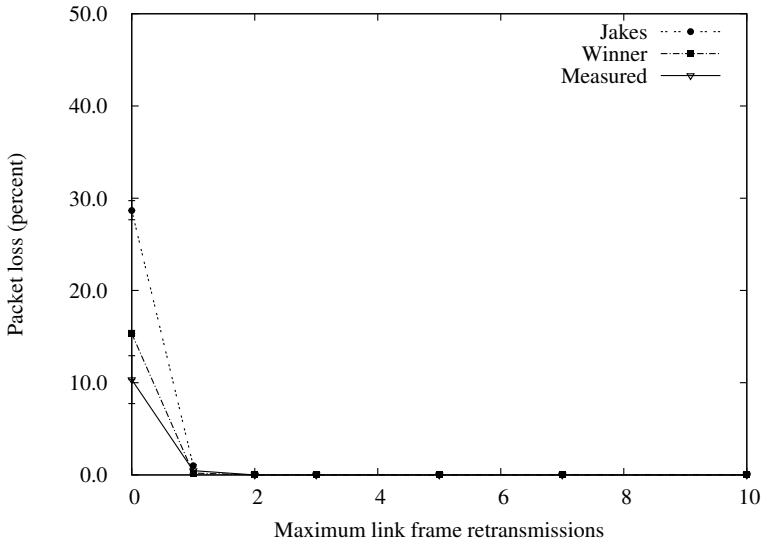


Figure 6.34: VoIP packet loss as a function of maximum allowed link layer retransmissions.

VoIP latency

VoIP latency experiments were performed by sending UDP packets with the previously mentioned VoIP packet characteristics (50 packets/s, 172 bytes payload per packet), and measuring the time difference from transmission to reception. The tools used were `mgen` [229] for traffic transmission and reception and `trpr` [229] for latency calculation.

The experiments were performed over the three different channels, replicated with the 30 channel realizations, with a range of link level retransmission delays and retransmission limits. The results are presented as graphs showing the cumulative delay distribution.

Figure 6.35 shows the delay distribution for the Jakes channel with a limit of 3 link retransmissions for each frame. On the x-axis there is the delay threshold, and on the y-axis is the amount of packets that have arrived within a specific delay.

At the left in the figure is a region of 10 ms where no packets have arrived. This represents the delay in the fixed network, which will

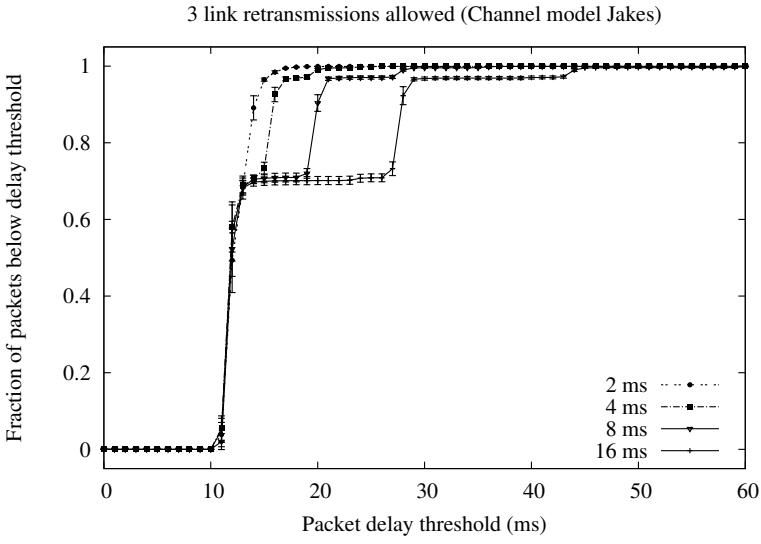


Figure 6.35: The cumulative delay distribution using the Jakes channel model and a maximum of 3 link frame retransmissions for different link retransmission delays.

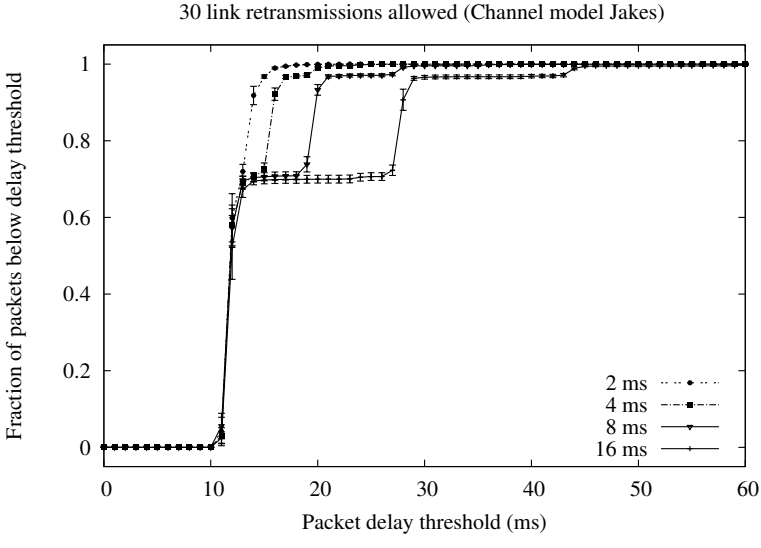


Figure 6.36: The cumulative delay distribution using the Jakes channel model and a maximum of 30 link frame retransmissions for different link retransmission delays.

give an offset (modeled as a fixed value) to the delay in the wireless network. Then there is a sharp rise for 3 ms within which about 70% of the packets arrive. These are packets with no retransmissions on the link layer. As the first, second and third retransmissions happen, more packets arrive (but with added delay). There is a slight slope for regions that would be believed to be fully vertical. This is due to a number of factors, which have been identified as clock drift between the sender and receiver, skewed transmission time slots (if a packet arrives too late to be transmitted in a timeslot and have to wait for the next), and inexactness of the traffic shaper that generates the fixed network delay (see for example [197]). Further, the experiments are repeated 30 times which also causes variation.

The horizontal regions in the figure correspond to different retransmission delays. This is most visible with the 16 ms retransmission delay: between 10-26 ms there is no packet delivery, since packets with a damaged link frame are queued awaiting a frame retransmission. After the first horizontal region there is another one between 26-42 ms that corresponds to the packets waiting for a second frame retransmission.

The experiments above were limited to a maximum of 3 link frame retransmissions. Figure 6.36 shows the results for a maximum of 30 link frame retransmissions. This means in practice "unlimited" retransmissions. Again, the result here display the same characteristics as in the previous figure. One notable feature is that there is not much difference between the two graphs. The explanation is found in the retransmission graph presented earlier; after about three retransmissions all frames have been successfully delivered.

For completeness, the results for the Winner and Measured channel are shown in Figures 6.37 – 6.40. They exhibit the same properties as were just discussed, but with different absolute numbers due to the channel characteristics. One notable difference in Figure 6.39 and 6.40 is the (relatively) large confidence interval mainly visible for 16 ms link retransmission delay. This was investigated and found to be caused by a single channel out of 30 that had about half the capacity of the others, causing the confidence interval to increase.

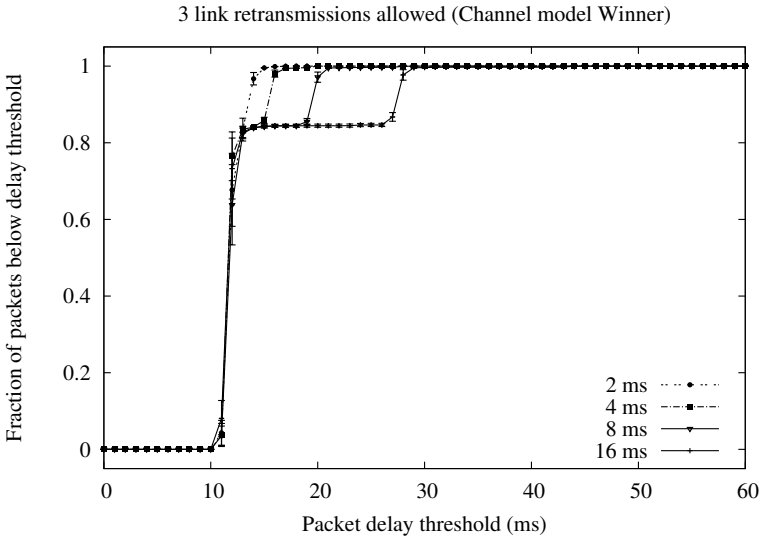


Figure 6.37: The cumulative delay distribution using the Winner channel model and a maximum of 3 link frame retransmissions for different link retransmission delays.

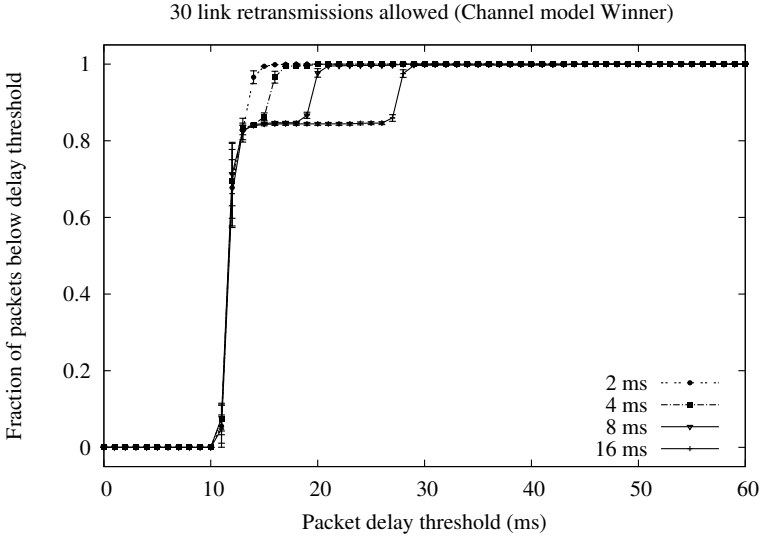


Figure 6.38: The cumulative delay distribution using the Winner channel model and a maximum of 30 link frame retransmissions for different link retransmission delays.

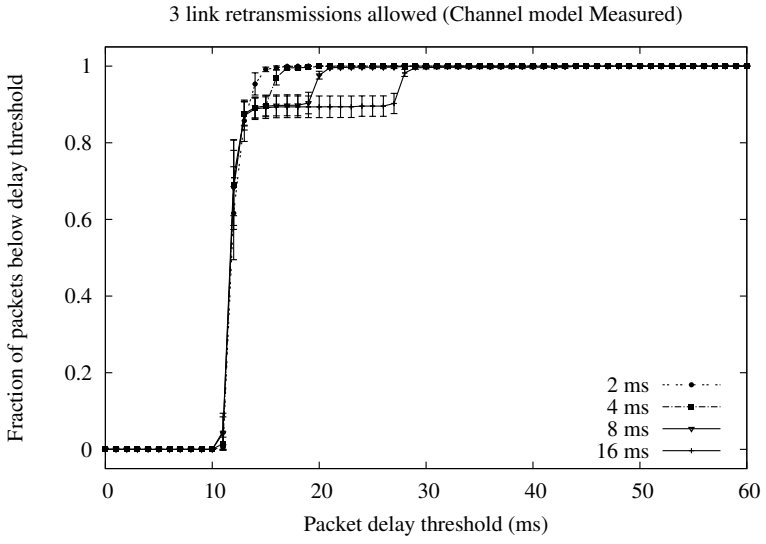


Figure 6.39: The cumulative delay distribution using the Measured channel model and a maximum of 3 link frame retransmissions for different link retransmission delays.

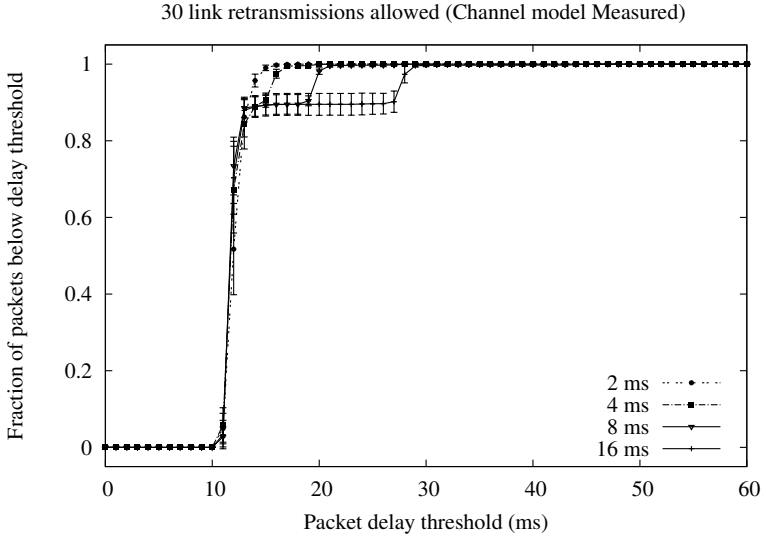


Figure 6.40: The cumulative delay distribution using the Measured channel model and a maximum of 30 link frame retranmissions for different link retranmission delays.

To summarize, these experiments show that the baseline Wireless IP parametrization supports VoIP network requirements with a large delay margin, and allows for link frame retransmissions to minimize packet loss caused by transmission errors.

To see the VoIP latency in more detail, Figure 6.41 shows a scatter plot of the latency of 1500 VoIP packets as a function of their transmission time, from one of the experiment runs with the Jakes channel model, 30 link frame retransmissions and 2 ms retransmission delay.

The plot shows clearly the 10 ms delay in the fixed network, after which the arriving packets have a varying latency distribution. One interesting artifact is the two thick lines, first at 12 and then 11 ms. This is due to the aforementioned clock drift. Packets are time stamped at the transmitter, and compared to the clock at the receiver. Although the clocks are synchronized at the start of each experiment run, the two clocks have drifted apart, which causes the displayed artifact. The 1 ms gap is due to the resolution of the latency calculations, so a drift much smaller than 1 ms may account for the difference due to rounding up or down.

After the majority of the packets have been received at 12 ms after the first transmission, there is another pronounced level at 14 ms where the second retransmission was successful, and the third retransmission (almost not intelligible), at 16 ms. Between these there are some packets with latencies in between. These are caused by missed timeslots and rounding effects as the latency calculation tool operated on integer milliseconds.

This behavior is even better displayed in Figure 6.42 which shows the latency scatter graph for the same experiment, but with 16 ms retransmission delay. Here the three retransmission levels are easily discernible.

6.4.3 Conclusions from application experiments

The results from the experiments show that the proposed system parametrization can support both ends of the application bandwidth-delay elasticity scale. As the parametrization of the adaptive modula-

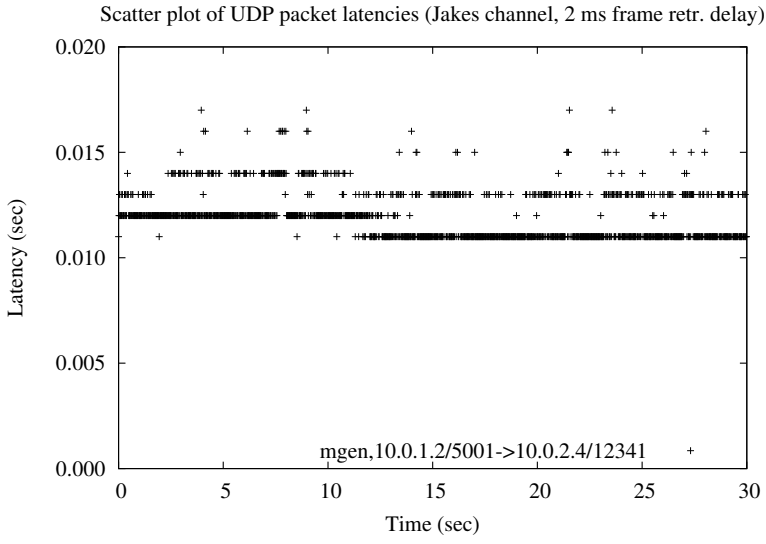


Figure 6.41: Scatter plot of UDP packet latencies for one specific experiment run with the Jakes channel model, 2 ms retransmission delay and unlimited retransmissions.

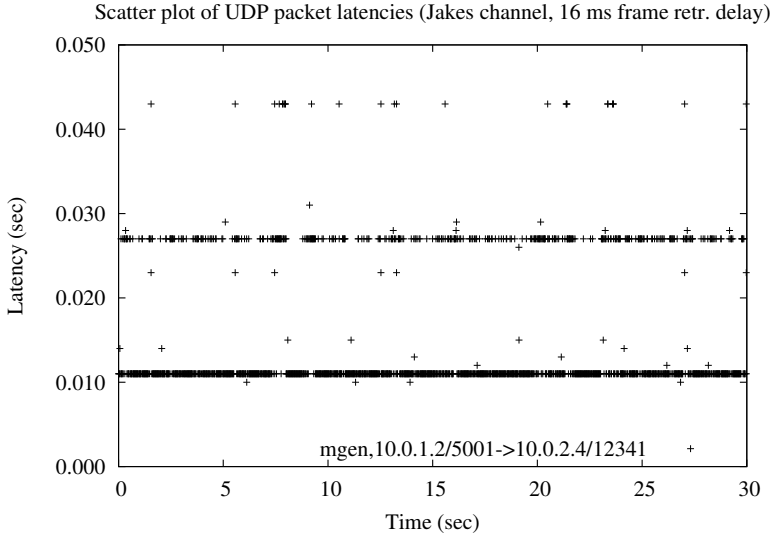


Figure 6.42: Scatter plot of UDP packet latencies for one specific experiment run with the Jakes channel model, 16 ms retransmission delay and unlimited retransmissions.

tion switch levels for this system were designed for bulk data transfer with unlimited retransmissions, it is interesting to see that the same settings also work well for applications with more stringent requirements on delay, and with good margin. This is in contrast with typical systems [2] where different traffic classes may receive differentiated link layer treatment (reliability/FEC/modulation), which adds to system complexity. The results indicate that a simpler scheme could provide the performance necessary, by using adaptive modulation and fast link retransmissions. Note that these results are based on a single-user and single-flow scenario. With multiple users and/or flows, the channel allocator/scheduler must still take flow requirements into consideration.

6.5 Experiment conclusions

This chapter has analyzed a number of factors that impact the transport layer performance.

- A first conclusion is a confirmation that the basic design of the Wireless IP downlink system proposal is sound. Performance gains analyzed at lower layers [69, 210] are achievable also at the transport layer.

- Using adaptive modulation as opposed to fixed modulation results in large TCP throughput improvements.

- A shorter retransmission delay results in better TCP throughput in the event of packet losses, and is also important to achieve a low transport packet delay for short flows and interactive real-time communication.

- Link frame retransmissions should not be limited. This is also in line with results by e.g. [138].

- By utilizing channel prediction and scheduled allocation, significant capacity gains can be achieved compared to using a static

channel allocation. There is also a multi-user gain where different users' experience different fading, but this has not been a topic in this dissertation. See [235] for analytical results.

- Further, buffer sizes should be sufficiently large for TCP to fully utilize the link capacity and to sustain throughput during packet loss events.

Some parameters are of lesser gain. The results indicate that the use of symbol combination gives very marginal TCP throughput improvements, although the link operates more efficiently. Optimizing the adaptive modulation switch tables for a target prediction error also showed marginal improvements. A positive aspect of the latter is that the used modulation switch tables were shown to be robust for a varying degree of channel prediction errors (NMSE 0-0.2), and having a single table decreases system complexity and computational requirements.

As a final conclusion, the tested applications, TCP data transfer and UDP real-time communications, perform well with the parameters of the Wireless IP downlink system proposal, with unlimited link retransmissions and short link retransmission delays.

Chapter 7

Concluding remarks

Today is a window,
tomorrow the landscape.
All you need to do is take a look outside,
to know what we are bound to face.

— Greg Graffin, *Markovian Process*

The topic of the dissertation is transport protocol performance in wireless networks, from a cross-layer perspective. The properties of the physical and link layers have been related to the performance impact on the transport and application layers in a number of ways. From the characteristics of radio communication, to surveying related work of TCP optimizations, to experimentally evaluating TCP and UDP under varying conditions in an emulated wireless system. This chapter contains a summary of the dissertation and concludes with ideas for future work.

7.1 Dissertation summary

Although the performance of Internet communication over wireless networks has been extensively studied for the last decades, there is still no general consensus or all encompassing solution. In this work,

the foundation of wireless communication and its effect on link layer and transport layer have been presented and discussed in detail.

The free space propagation of wireless radio transmission, compared to wired transmission, leads to an increased variability in the received signal. Multi-path fading, shadow fading, interference, attenuation and other effects distort transmitted signals. This may lead to frame and packet loss in the wireless network. This in turn causes problems for TCP, which assumes packet loss due to damage is rare, and that congestion is the cause of loss.

To alleviate this, there exist a number of proposals. One group of studies suggests modifying both sender and receiver, to implement a more robust capacity estimation that is not sensitive to non-congestion related packet loss, or to add options to increase the information exchange between the end nodes. Other proposals focus on hiding the problems that occur in the wireless link. By doing persistent link-level retransmissions, there is no packet loss that can be misclassified. If link retransmissions are slow, this can instead lead to problematic delays. Other proponents take an approach where the network notifies when transmission problems occur. A base station that is aware of the channel conditions can signal this to the end hosts, for example. A fourth group of approaches split the transmission: one connection over the wired network, and a second connection over the wireless network. This isolates the problems of the wireless part to the same.

An idea put forward in this dissertation is to relax the requirements of full TCP reliability and accept bit errors. With the TCP-L modification, a receiver that is able to use possibly damaged data can allow delivery of corrupt TCP segments in exchange for higher throughput. As these corrupt segments are acknowledged, the TCP sender does not invoke congestion control and a higher throughput is achieved compared to a non-modified TCP. This implies a degree of uncertainty, as the header must be recovered from possible bit errors. There is also uncertainty as to which TCP connection of a host a corrupted segment belongs. By making a few assumptions about the connections, the sensitive part of the header is minimized to the sequence number, if a matching connection can be found. It

is suggested that only packets that have matching port numbers and sequence numbers to existing connections are used, to minimize the risk of inserting the data in the wrong connection. Other packets can safely be dropped and will be handled like a normal packet loss, with the resulting retransmission and initiation of congestion control at the sender. The modification has been implemented in the Linux kernel, and is shown to perform well in experimental evaluations. Results show that even with a small amount of residual bit errors, a large throughput improvements can be made over standard TCP. Seen from another perspective, TCP-L upholds the sending rate instead of reducing it in response to perceived congestion.

Another method to improve the transport layer performance is to provide a more reliable wireless connection. This reduces packet loss due to link errors. To this end, the downlink part of the Wireless IP evaluation system has been validated in a system emulation. The characteristic features of the system, with persistent and fast link level retransmissions on a short time scale, adaptive modulation and scheduling based on channel predictions over OFDM channels are shown to provide a good service for both bulk data transfer as well as interactive real-time traffic[1]. The reasons for this is that even if channel errors are exist, with fast link retransmission (on the time scale of milliseconds) transmission errors can be recovered without introducing excessive delays. Concerns have been raised about the effect of link rate variability on TCP throughput. With the adaptive modulation on very short time scales, the rate was very much varying on the link. However, this was not found to cause negative effects on TCP. The reasons for this is that TCP kept the link buffer utilized, meaning there were always packets available for transmission. This is also due to the persistent link retransmissions. As TCP did not experience non-congestion related packet loss, there was no reduction of the sending rate that could cause empty transmit buffers. It should be noted that other TCP implementations may perform differently. E.g., [147] showed measurements where TCP Cubic was aggressively filling buffers, while TCP Vegas was more conservative.

[1]It should be noted that these were examined in isolation, and the effects of competing traffic were not investigated.

7.2 Future work

By now, some of the wireless link design considerations that have been examined and evaluated in this dissertation, are deployed commercially in 3GPP LTE. It would therefore be interesting to perform measurements in a live network, using the same applications and evaluation methodology, and compare to the results from the system emulation. It should be straightforward to exchange the emulated system with a live system, and re-execute the experiments. Preferably this should be done in cooperation with a network operator to have better control, or at least more information, of the parameters that are not observable from the mobile terminal. For example statistics on the amount of users in the cell and channel quality indications. Further, 3GPP LTE-Advanced is being implemented, and it would be interesting to perform system measurements at an early stage. For this system, *cooperative multipoint transmission* (CoMP) is currently being researched [55]. For example, one aspect of CoMP uses joint transmission from different base stations to terminals. With combined signals, the received signal strength can be increased and correspondingly attain a higher capacity or better resiliency.

With regards to TCP-L, it aims to increase throughput to reduce data delivery delay to the user, at the expense of reliability. As discussed, it is better to do local link retransmissions if the link retransmission delay is short. This depends in part on the propagation delay. Therefore an interesting scenario for TCP-L could be in e.g. satellite or long-range communication systems, where the retransmission delay can be large compared to the end-to-end delay.

□

Appendix A

Experiment parameter overview

This appendix provides an overview of the parameters used in the experiments in Chapter 6. The parameters were outlined in Table 5.8, and the specific settings for each experiment in Chapter 6 is detailed here. The experiments are divided in three groups (A, B, C). Group A contains experiments with base line parameters, presented in Section 6.2. Group B contains experiments with varying link and network parameters, presented in Section 6.3. Group C contains application level experiments, presented in Section 6.4. These groups are further divided into the items below. Table A.1 shows a matrix of all configurable parameters, the parameter range, the default setting, and the actual value or value range used for a specific experiment. The table cells of a parameter/experiment crossing show the values that are different from the default setting. An empty value for a cell then represents the default value. The cell mark n/a indicates that the cell value is not applicable. It is used for the application parameters, where only one application type is used in each specific experiment.

A1 Link retransmissions vs fixed vs adaptive modulation.
A2 Soft combining/Chase combining.
A3 Varying channel prediction error.
A4 Channel allocation strategies.
A5 Distribution of modulation levels.
B1 Varying link retransmission delay.
B2 Varying delay in fixed network.
B3 Packet loss in fixed network.
B4 Effect of buffer sizing.
C1 Varying TCP flow lengths.
C2 Voice over IP.

	Parameter range	Default setting	A1	A2	A3	A4	A5	B1	B2	B3	B4	C1	C2
Channel data													
Channel	Jakes, Winner, Measured	J., W., M.											
Channel predictor NMSE	0.0-0.2	0.1	0.0	0.0	0-0.2		0.0/0.1						
Channel allocation	scheduled/static	scheduled	static	static	static	s/s	s/s						
Bins per timeslot	0-25	10											
Frame													
Frame size	108 symbols	108											
Modulation	fixed/adaptive	adaptive	f/a										
Adaptive modulation switch levels	max throughput, NMSE 0, 0.1	NMSE 0.1	0.0	0.0	0.0/0.1		0.0/0.1						
Symbol/frame error calc	awgn, awgn+soft combining	a+sc	a	a/a+sc									
Retransmissions	0-30	30	1-30										0-30
Chase combining	on/off	on	off	on/off									
Deliver err. frames	on/off	on											
Retr delay	2-16 ms	2						2-16		2-16	2-16		2-16
Ordering	on/off	off											
Packet													
Queueing/buffering	1-54 packets	54									1-20		
Corrupt packets	discard/deliver	deliver											
Delay in fixed net	0-150 ms	10							0-150			0-50	
Loss in fixed net	0-10 %	0								0-10			
Ordering	on/off	on											
Application													
Data download	TCP, varying flow length	30 sec flow										1-10000 kB	n/a
VoIP	UDP, typical RTP 50 pkt/s, 172 bytes/pkt	50 p/s, 172 b/p	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	50 p/s, 172 b/p

Table A.1: Experiment parameter overview for all experiments.

Bibliography

[1] 3GPP. TS 25.853: V4.0.0 (2001-03). Technical Specification Group (TSG) RAN; Delay Budget within the Access Stratum, 2001. Available at `http://www.3gpp.org/ftp/Specs/html-info/25853.htm`.

[2] 3GPP. Digital cellular telecommunications system (phase 2+); UMTS; LTE; Quality of service (QoS) concept and architecture (3GPP TS 23.107 version 10.0.0 release 10). *ETSI Technical Specification*, Mar. 2011.

[3] Adobe Corporation. Flash player. `http://www.adobe.com/products/flashplayer/` Visited 2011-09-27.

[4] J. Alcaraz and F. Cerdan. Improving TCP performance over 3G links with an ACK rate control algorithm. *Proceedings of the International Symposium of Wireless Communication Systems (ISWCS)*, pages 480–484, 6-8 Sept. 2006.

[5] S. Alfredsson. TCP in wireless networks: Challenges, optimizations and evaluations. *Licentiate thesis, Karlstad University, Sweden*, 2005.

[6] S. Alfredsson and A. Brunstrom. TCP-L: Allowing bit errors in wireless TCP. *Proceedings of IST Mobile and Wireless Communications Summit 2003*, June 2003.

[7] S. Alfredsson, A. Brunstrom, and M. Sternad. A 4G link level emulator for transport protocol evaluation. In *Proceedings of the Swedish National Computer Networking Workshop (SNCNW)*, Karlstad, Sweden, Nov. 2004.

[8] S. Alfredsson, A. Brunstrom, and M. Sternad. Emulation and validation of a 4G system proposal. In *Proceedings of Radiovetenskap och Kommunikation*, Linkoping, Sweden, June 2005.

[9] S. Alfredsson, A. Brunstrom, and M. Sternad. Transport protocol performance over 4G links: Emulation methodology and results. In *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS)*, Valencia, Spain, Sept. 2006.

[10] S. Alfredsson, A. Brunstrom, and M. Sternad. Cross-layer analysis of TCP performance in a 4G system. In *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS)*, Reykjavik, Iceland, Oct. 2007.

[11] S. Alfredsson, A. Brunstrom, and M. Sternad. Impact of 4G wireless link configurations on VoIP network performance. In *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Dubrovnik, Croatia, Sept. 2007.

[12] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's loss recovery using limited transmit. RFC 3042 (Proposed Standard), Jan. 2001.

[13] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's initial window. RFC 3390 (Proposed Standard), Oct. 2002.

[14] M. Allman and S. Ostermann. ONE: The Ohio Network Emulator. Technical Report TR-19972, Ohio University, Aug. 1997.

[15] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (Draft Standard), Sept. 2009.

[16] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581 (Proposed Standard), Apr. 1999.

[17] L. Andrew, T. Cui, J. Sun, M. Zukermanber, K.-T. Ko, and S. Chan. Buffer sizing for nonhomogeneous TCP sources. *IEEE Communication letters*, 9:6:567–569, June 2005.

[18] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proceedings of ACM SIGCOMM*, pages 281–292, 2004.

[19] D. Aronsson. *Channel Estimation and Prediction for MIMO OFDM Systems*. PhD thesis, Dept. Engineering Sciences, Signals and Systems, Uppsala University, Sweden, 2011.

[20] M. Assaad and D. Zeghlache. Cross-layer design in HSDPA system to reduce the TCP effect. *IEEE Journal on Selected Areas in Communications*, 24(3):614–625, Mar. 2006.

[21] A. Baiocchi, A. P. Castellani, and F. Vacirca. YeAH-TCP: Yet Another Highspeed TCP. In *Proceedings of Fifth International Workshop on Fast Long Distance Networks (PFLDnet)*, Los Angeles, USA, 2007.

[22] A. V. Bakre and B. Badrinath. Implementation and performance evaluation of Indirect TCP. *IEEE Transactions on Computers*, 46(3), March 1997.

[23] B. 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*, Baltimore, USA, May 1997.

[24] H. Balakrishnan and R. Katz. Explicit loss notification and wireless web performance. In *Proceedings Globecom Internet Mini-Conference*, Sydney, Australia, November 1998.

[25] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *Proceedings of the First Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Berkeley, CA, USA, Nov. 1995.

[26] R. K. Balan, B. P. Lee, K. R. R. Kumar, L. Jacob, W. K. G. Seah, and A. L. Ananda. TCP HACK: TCP header checksum option to improve performance over lossy links. In *Proceedings of 20th IEEE Conference on Computer Communications (INFOCOM)*, Anchorage, Alaska, USA, Apr. 2001.

[27] M. Bateman and S. Bhatti. TCP testing: How well does ns2 match reality? In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 276–284, Perth, Australia, Apr. 2010.

[28] C. Bettstetter, H.-J. Vögel, and J. Eberspächer. GSM phase 2+ general packet radio service GPRS: Architecture, protocols, and air interface. *IEEE Communication Surveys*, 2(3), 1999.

[29] S. Bhandarkar, A. L. N. Reddy, M. Allman, and E. Blanton. Improving the robustness of TCP to non-congestion events. RFC 4653 (Experimental), Aug. 2006.

[30] S. Bhandarkar, N. E. Sadry, A. N. Reddy, and N. H. Vaidya. TCP-DCR: A novel protocol for tolerating wireless channel errors. In *IEEE Transactions on Mobile Computing*, volume 4, pages 517–529, Sept. 2005.

[31] S. Biaz and N. Vaidya. Discriminating congestion losses from wireless losses using inter-arrival times at the receiver. In *Proceedings of the IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (ASSET)*, Richardson, TX, USA, Mar. 1999.

[32] S. Biaz and N. H. Vaidya. 'De-randomizing' congestion losses to improve TCP performance over wired-wireless networks. *IEEE/ACM Transactions on Networking*, 13(3):596–608, 2005.

[33] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475 (Informational), Dec. 1998.

[34] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Performance enhancing proxies intended to mitigate link-related degradations. RFC 3135 (Informational), June 2001.

[35] R. Braden. Requirements for Internet hosts - communication layers. RFC 1122 (Standard), Oct. 1989.

[36] 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*, 13(8):1465–1480, Oct. 1995.

[37] A. Brunstrom, K. Asplund, and J. Garcia. Enhancing TCP performance by allowing controlled loss. In *Proceedings of the SSGRR 2000 Computer & eBusiness Conference*, L'Aquila, Italy, Aug. 2000.

[38] P. Bucknell and M. Baker. Cross-layer interactions in UMTS evolved UTRAN (E-UTRAN). In *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS)*, pages 262–266, Oct. 2007.

[39] R. Burnett, A. Brunstrom, and A. Nilsson. *Perspectives on Multimedia - Communication, Media and Information Technology.* Wiley, 2004. ISBN 0-470-86863-5.

[40] R. Bush and D. Meyer. Some Internet architectural guidelines and philosophy. RFC 3439 (Informational), Dec. 2002.

[41] C. Caini and R. Firrincieli. TCP Hybla: a TCP enhancement for heterogeneous networks. *International Journal of Satellite Communications and Networking*, 22, 2004.

[42] M. Carbone and L. Rizzo. Dummynet revisited. *SIGCOMM Compututer Communcation Review*, 40:12–20, Apr. 2010.

[43] B. Carpenter. Architectural principles of the Internet. RFC 1958 (Informational), June 1996.

[44] J. Case, M. Fedor, M. Schoffstall, and J. Davin. Simple network management protocol (SNMP). RFC 1157 (Historic), May 1990.

[45] T. Chahed, A.-F. Canton, and S.-E. Elayoubi. End-to-end TCP performance in W-CDMA / UMTS. In *Proceeings of the IEEE International Conference on Communications (ICC)*, volume 1, pages 71–75, May 2003.

[46] R. Chakravorty, J. Cartwright, and I. Pratt. Practical experience with TCP over GPRS. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Taipei, Taiwan, Nov. 2002.

[47] M. C. Chan and R. Ramjee. TCP/IP performance over 3G wireless links with rate and delay variation. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Atlanta, USA, Sept. 2002.

[48] M. C. Chan and R. Ramjee. TCP/IP performance over 3G wireless links with rate and delay variation. *Wireless Networks*, 11(1-2):81–97, 2005.

[49] M. C. Chan and R. Ramjee. Improving TCP/IP performance over third-generation wireless networks. *IEEE Transactions on Mobile Computing*, 7(4):430–443, 2008.

[50] D. Chase. Code combining–a maximum-likelihood decoding approach for combining an arbitrary number of noisy packets. *IEEE Transactions on Communications*, 33(5):385–393, 1985.

[51] K. Chebrolu, B. Raman, and R. R. Rao. A network layer approach to enable TCP over multiple interfaces. *Wireless Networks*, 11(5):637–650, 2005.

[52] W.-P. Chen, Y.-C. Hsiao, J. C. Hou, Y. Ge, and M. P. Fitz. Syndrome: a light-weight approach to improving TCP performance in mobile wireless networks. *The Journal of Wireless Communications and Mobile Computing*, pages 37–57, 2002.

[53] F. Chengpeng. *TCP Veno: End-To-End Congestion Control Over Heterogeneous Networks*. PhD thesis, Chinese University of Hong Kong, July 2001.

[54] L. Cui, S. J. Koh, X. Cui, and Y. J. Kim. Adaptive increase and adaptive decrease algorithm for wireless TCP. In *International Conference on Natural Computation (ICNC)*, volume 2, pages 392–398, Hainan, China, Aug. 2007.

[55] E. Dahlman, S. Parkvall, and J. Sköld. *4G LTE/LTE-Advanced for Mobile Broadband*. Academic Press, 2011. ISBN 978-0-12-385489-6.

[56] D. F. B. de Oliveira, A. Munaretto, A. Ziviani, and M. Fonseca. A proxy-based architecture for TCP to mitigate packet loss on wireless networks. In *Proceedings of the 2nd IFIP conference on Wireless Days*, pages 332–337, Piscataway, NJ, USA, 2009.

[57] C. Deák, L. Farkas, G. Hományi, and C. Vulkán. Downlink streaming performance over evolved HSPA. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing (IWCMC)*, pages 1035–1039, Leipzig, Germany, 2009.

[58] S. Deering and R. Hinden. Internet Protocol, version 6 (IPv6) specification. RFC 2460 (Draft Standard), Dec. 1998.

[59] M. Degermark, M. Engan, B. Nordgren, and S. Pink. Low-loss TCP/IP header compression for wireless networks. *Wireless Networks*, 3(5):375–387, Oct. 1997.

[60] R. Dettmer. The convergent phone. *IEE Review*, 48(1):23–27, Jan. 2002.

[61] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *SIGCOMM Computer Communication Review*, 36:59–62, Jan. 2006.

[62] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. An argument for increasing TCP's initial congestion window. *ACM SIGCOMM Computer Communication Review*, 40:26–33, June 2010.

[63] T. Ekman, M. Sternad, and A. Ahlén. Unbiased power prediction on broadband channels. In *IEEE Vehicular Technology Conference (VTC-Fall)*, Vancouver, Canada, Sept. 2002.

[64] H. El-Ocla. TCP CERL: congestion control enhancement over wireless networks. *Wireless Networks*, 16:183–198, January 2010.

[65] H. Elaarag. Improving TCP performance over mobile networks. *ACM Computing Surveys*, 34(3):357–374, August 2002.

[66] ETSI. Digital cellular telecommunications system (Phase 2+) (GSM); Enhanced Full Rate (EFR) speech transcoding (GSM 06.60 version 8.0.1 Release 1999). *ETSI Technical Specification*, Nov. 2010.

[67] J. Faezah and K. Sabira. Adaptive modulation for OFDM systems. *International Journal of Communication Networks and Information Security (IJCNIS)*, 1(2), 2009.

[68] G. Fairhurst and L. Wood. Advice to link designers on link automatic repeat request (ARQ). RFC 3366 (Best Current Practice), Aug. 2002.

[69] S. Falahati, A. Svensson, T. Ekman, and M. Sternad. Adaptive modulation systems for predicted wireless channels. *IEEE Transactions on Communications*, pages 307–316, 2004.

[70] S. Falahati, A. Svensson, M. Sternad, and H. Mei. Adaptive trellis-coded modulation over predicted flat fading channels. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall)*, Orlando, Florida, USA, Oct. 2003.

[71] S. Floyd. Highspeed TCP for large congestion windows. RFC 3649 (Experimental), Dec. 2003.

[72] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification. RFC 5348 (Proposed Standard), Sept. 2008.

[73] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard), Apr. 2004.

[74] S. Floyd and E. Kohler. Internet research needs better models. *SIGCOMM Computer Communication Review*, 33:29–34, January 2003.

[75] S. Floyd and E. Kohler. Profile for datagram congestion control protocol (DCCP) congestion control id 2: TCP-like congestion control. RFC 4341 (Proposed Standard), Mar. 2006.

[76] S. Floyd and V. Paxson. Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, Aug. 2001.

[77] J. D. Foley. *Computer graphics: principles and practice.* Addison-Wesley Systems Programming Series, 1997.

[78] Y. Ganjali. *Buffer sizing in Internet routers.* PhD thesis, Electrical engineering, Stanford University, Mar. 2007.

[79] Y. Ganjali and N. McKeown. Update on buffer sizing in Internet routers. *ACM SIGCOMM Computer Communications Review*, 36:5:67–70, Oct. 2006.

[80] D. Gao, Y. Shu, L. Yu, M. Y. Sanadidi, and M. Gerla. TCP SPC: Statistic process control for enhanced transport over wireless links. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 5453–5457, New Orleans, USA, 2008.

[81] J. Garcia, S. Alfredsson, and A. Brunstrom. The impact of loss generation on emulation-based protocol evaluation. In *Proceedings International Conference on Parallel and Distributed Computing and Networks (PDCN)*, Innsbruck, Austria, Feb. 2006.

[82] J. Garcia and A. Brunstrom. A robust JPEG coder for a partially reliable transport service. In *Proceedings 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS)*, Enschede, The Netherlands, Oct. 2000. Springer.

[83] J. Garcia and A. Brunstrom. Checksum-based loss differentiation. In *Proceedings 4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*, Stockholm, Sweden, Sept. 2002.

[84] J. Garcia, E. Conchon, T. Perennou, and A. Brunstrom. KauNet: improving reproducibility for wireless and mobile research. In *Proceedings of the 1st international workshop on System evaluation for mobile platforms*, pages 21–26, San Juan, Puerto Rico, 2007.

[85] J. Garcia, P. Hurtig, and A. Brunstrom. KauNet: A versatile and flexible emulation system. In *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW)*, Apr. 2008.

[86] J. T. Gibbs. The FreeBSD Project, 2011. `http://www.freebsd.org` Visited 2012-03-14.

[87] A. Goel, C. Krasic, K. Li, and J. Walpole. Supporting low latency TCP-based media streams. In *Proceedings of the Tenth International Workshop on Quality of Service (IWQoS)*, Miami Beach, Florida, USA, May 2002.

[88] S. Goel and D. Sanghi. Improving TCP performance over wireless links. In *Proceedings of the IEEE Region Ten Conference on Global Connectivity in Energy, Computer Communication and Control (TENCON)*, New Delhi, India, Dec. 1998.

[89] T. Goff, J. Moronski, D. Phatak, and V. V. Gupta. Freeze-TCP: A true end-to-end enhancement mechanism for mobile environments. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, March 2000.

[90] A. Goldsmith. *Wireless Communications*. Cambridge University Press, Cambridge, New York, 2005. ISBN 978-0-521-83716-3.

[91] H. Granbohm and J. Wiklund. GPRS - general packet radio service. *Ericsson Review*, 2:82–88, 1999.

[92] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control. *ACM Computer Communication Review*, pages 25–38, 2004.

[93] K.-J. Grinnemo, J. Garcia, and A. Brunstrom. Taxonomy and survey of retransmission based partially reliable transport protocols. *Computer Communications*, 27(15):1441–1452, 2004.

[94] A. Gurtov and R. Ludwig. Responding to spurious timeouts in TCP. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 3, pages 2312–2322, 2003.

[95] S. Ha, I. Rhee, and L. Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 42(5):64–74, 2008.

[96] U. Herzog. WINNER II Final Report (IST-4-027756 D7.1.5 v1.0), Feb. 2008. Available online at `http://www.ist-winner.org/deliverables.html` . Visited 2011-10-10.

[97] H. Holma, M. Kuusela, E. Malkamaki, K. Ranta-aho, and C. Tao. VOIP over HSPA with 3GPP release 7. In *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Helsinki, Finland, Sept. 2006.

[98] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda. Is it still possible to extend TCP? In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement*, IMC '11, pages 181–194, New York, NY, USA, 2011. ACM.

[99] K. Hoque, R. Haque, M. Hossain, M. Farazi, and G. Hossain. Modeling and performance of TCP in a MC-CDMA system for 4G communications. In *Proceedings of the 10th International Conference on Computer and Information Technology (ICCIT)*, Dec. 2007.

[100] E. Hossain, D. I. Kim, and V. Bhargava. TCP performance under dynamic link adaptation in cellular multi-rate WCDMA networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 3, pages 1834–1839, 2002.

[101] P. Hurtig. *Transport-Layer Performance for Applications and Technologies of the Future Internet*. PhD thesis, Dept. of Computer Science, Karlstad University, Sweden, Feb. 2012. Karlstad University Studies 2011:65, ISBN 978-91-7063-404-8.

[102] IEEE LAN/MAN Standards Committee. IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2007*, June 2007.

[103] IEEE LAN/MAN Standards Committee. IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009*, 2009.

[104] D. B. Ingham and G. D. Parrington. Delayline: A wide-area network emulation tool. *Computing Systems*, 7(3):313–332, 1994.

[105] International Telecommunication Union. About ITU, 2012. `http://www.itu.int/en/about/Pages/default.aspx`. Visited 2012-03-12.

[106] ISO. *ISO/IEC 7498:1984: Information processing systems – Open Systems Interconnection – Basic Reference Model*. International Organization for Standardization, 1984.

[107] ITU-D. ITU statistics; Internet users per 100 inhabitants 2010. Available at `http://www.itu.int/ict/statistics` Visited 2011-09-27.

[108] ITU ICT Data and Statistics Division. *Measuring the Information Society*. International Telecommunication Union, Sept. 2011. Available at `http://www.itu.int/pub/D-IND-ICTOI-2011/en` or `http://www.itu.int/ITU-D/ict/publications/idi/2011/Material/MIS_2011_without_annex_5.pdf` Visited 2011-10-11.

[109] ITU-R. *Report ITU-R M.2135-1: Guidelines for evaluation of radio interface technologies for IMT-Advanced*. ITU-R, 2009. Available at `http://www.itu.int/pub/R-REP-M.2135-1-2009` Visited 2012-03-14.

[110] ITU-T. Recommendation G.711 - pulse code modulation (PCM) of voice frequencies. *ITU-T*, 1988.

[111] ITU-T. *Recommendation P.800: Methods for objective and subjective assessment of quality*. ITU-T, 1996.

[112] ITU-T. *Recommendation G.107: The E-Model, a computational model for use in transmission planning*. ITU-T, 2000.

[113] ITU-T. *Recommendation P.862: Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs*. ITU-T, 2001.

[114] V. Jacobson. Compressing TCP/IP headers for low-speed serial links. RFC 1144 (Proposed Standard), Feb. 1990.

[115] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323 (Proposed Standard), May 1992.

[116] V. Jacobson and M. J. Karels. Congestion avoidance and control. In *Proceedings of SIGCOMM '88 Workshop*, pages 314–329, 1988.

[117] R. Jain. The art of computer systems performance analysis. *John Wiley & Sons, Inc.*, 1991.

[118] W. C. Jakes. *Microwave mobile communications*. IEEE Press, Piscataway, N.J., USA, 1994.

[119] R. A. Jones. Netperf. Available at `http://www.netperf.org` Visited 2011-12-28.

[120] H. Jung, N. Choi, Y. Seok, T. Kwon, and Y. Choi. Augmented Split-TCP over wireless LANs. In *IEEE International Conference on Communications (ICC)*, volume 12, pages 5420–5425, June 2006.

[121] M. Jurvansuu, J. Prokkola, M. Hanski, and P. Perala. HSDPA performance in live networks. In *Proceedings of the IEEE International Conference Communications (ICC)*, pages 467–471, June 2007.

[122] T. Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. *ACM Computer Communication Review*, 32(2), Apr. 2003.

[123] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401 (Proposed Standard), Nov. 1998.

[124] T. Klein, K. Leung, R. Parkinson, and L. Samuel. Avoiding spurious TCP timeouts in wireless networks by delay injection. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, volume 5, pages 2754–2759, Dallas, Texas, USA, 2004.

[125] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP). RFC 4340 (Proposed Standard), Mar. 2006.

[126] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen. Seawind: A wireless network emulator. In *Proceedings of 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems*, September 2001.

[127] J. Koo, S.-G. Mun, and H. Choo. Sender-based TCP scheme for improving performance in wireless environment. In *Proceedings of the International Conference on Computational Science (ICCS)*, volume 4490 of *Lecture Notes in Computer Science*, pages 538–541. Springer, 2007.

[128] J. Korhonen and P. Frossard. Bit-error resilient packetization for streaming H.264/AVC video. In *Proceedings of the International workshop on mobile video*, pages 25–30, Augsburg, Bavaria, Germany, Sept. 2007.

[129] C. Krasic, K. Li, and J. Walpole. The case for streaming multimedia with TCP. *International Workshop on Interactive Distributed Multimedia Systems (IDMS)*, Sept. 2001.

[130] R. Krishnan, M. Allman, C. Partridge, and J. P. Sterbenz. Explicit transport error notification for error-prone wireless and satellite networks. *BBN Technical Report No. 8333, BBN Technologies*, Mar. 2002.

[131] P. Kulkarni, M. Sooriyabandara, and L. Li. Improving TCP performance in wireless networks by classifying causes of packet losses. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2009.

[132] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet, 4th ed.* Addison Wesley, 2008. ISBN 978-0-321-49770-3.

[133] A. Kuzmanovic and E. W. Knightly. TCP-LP: low-priority service via end-point congestion control. *IEEE/ACM Transactions on Networking*, 14:739–752, Aug. 2006.

[134] C. Lai, K.-C. Leung, and V. Li. Enhancing wireless TCP: A serialized-timer approach. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, Mar. 2010.

[135] S. Landström. *TCP/IP Technology for Modern Network Environments.* PhD thesis, Luleå University of Technology, Dept. of Computer Science and Engineering, 2008. Publication 2008:30, ISSN 1402-1544.

[136] L.-Å. Larzon, M. Degermark, and S. Pink. UDP Lite for real-time multimedia applications. *Proceedings of the IEEE International Conference of Communications (ICC) - Quality of Service Mini Conference*, June 1999.

[137] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, and G. Fairhurst. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828 (Proposed Standard), July 2004.

[138] F. Lefevre and G. Vivier. Optimizing UMTS link layer parameters for a TCP connection. In *Proceedings of IEEE Vehicular Technology Conference (VTC-spring)*, volume 4, pages 2318–2322, May 2001.

[139] D. Leith and R. Shorten. H-TCP: TCP for high-speed and long-distance networks. In *Proceedings of the 2nd Workshop on Protocols for Fast Long Distance Networks (PFLDnet)*, Argonne, USA, Feb. 2004.

[140] K. Leung, T. Klein, C. Mooney, and M. Haner. Methods to improve TCP throughput in wireless networks with high delay variability. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-fall)*, volume 4, pages 3015–3019, Sept. 2004.

[141] V. Li and Z.-Q. Liu. PET: enhancing TCP performance over 3G beyond networks. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall)*, volume 4, pages 2302–2306, Oct. 2003.

[142] C. Lim and J. Jang. An adaptive end-to-end loss differentiation scheme for TCP over wired/wireless networks. *IJCSNS International Journal of Computer Science and Network Security*, 7(3):72–83, March 2007.

[143] H. Lin and S. Das. Performance study of TCP/RLP/MAC in next generation CDMA systems. In *Proceedings of the IEEE Personal, Indoor and Mobile Radio Communications conference (PIMRC)*, volume 1, pages 648–652, Sept. 2003.

[144] S. Lin and D. J. Costello. *Error Control Coding, Second Edition*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[145] Y.-B. Lin and I. Chlamtac. *Wireless and Mobile Network Architectures*. John Wiley & Sons, Inc., 2000.

[146] S. Liu, T. Başar, and R. Srikant. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Performance Evaluation*, 65:417–440, June 2008.

[147] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3G network: interplay between the wireless channel and applications. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking(MOBICOM)*, pages 211–222, San Francisco, California, USA, 2008.

[148] E. Lochin, T. Pérennou, and L. Dairaine. When should I use network emulation? *Annals of Telecommunications*, July 2011. (online pre-print, doi:10.1007/s12243-011-0268-5).

[149] M. López-Benítez, F. Bernardo, N. Vučević, and A. Umbert. Real-time emulation of heterogeneous wireless networks with end-to-edge quality of service guarantees: the AROMA testbed. *EURASIP Journal on Wireless Communication and Networking*, 2010:22:1–22:12, Apr. 2010.

[150] C. Lott, O. Milenkovic, and E. Soljanin. Hybrid ARQ: Theory, state of the art and future directions. In *Proceedings of the IEEE Workshop on Information Theory for Wireless Networks*, pages 1–5, July 2007.

[151] R. Ludwig and R. H. Katz. The Eifel algorithm: Making TCP robust against spurious retransmissions. *ACM Computer Communication Review*, 30(1):30–37, Jan. 2000.

[152] R. Ludwig and M. Meyer. The Eifel detection algorithm for TCP. RFC 3522 (Experimental), Apr. 2003.

[153] T. Mahmoodi, V. Friderikos, O. Holland, and A. H. Aghvami. Cross-layer design to improve wireless TCP performance with link-layer adaptation. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-fall)*, pages 1504–1508, Sept. 2007.

[154] M. Malkowski and S. Heier. Interaction between UMTS MAC scheduling and TCP flow control mechanisms. In *Proceedings of 2003 International Conference on Communication Technology*, pages 1373–1376, Beijing, China, Apr 2003.

[155] C. Marin, Y. Leprovost, M. Kieffer, and P. Duhamel. Robust MAC-lite and soft header recovery for packetized multimedia transmission. *IEEE Transactions on Communication*, 58:775–784, March 2010.

[156] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018 (Proposed Standard), Oct. 1996.

[157] S. McCanne and S. Floyd. VINT Network Simulator - ns (version 2). *http://isi.edu/nsnam/ns/*, April 1999.

[158] D. McCreary, K. Li, S. A. Watterson, and D. K. Lowenthal. TCP-RC: a receiver-centered TCP protocol for delay-sensitive applications. In *Proceedings of Multimedia Computing and Networking*, pages 126–130, San Jose, CA, USA, 2005. SPIE.

[159] J. Meinilä, T. Jämsä, P. Kyösti, D. Laselva, H. El-Sallabi, J. Salo, C. Schneider, and D. Baum. Determination of propagation scenarios. WINNER Deliverable D5.2, IST-2003-507581 WINNER, 2004. Available at `http://www.ist-winner.org/DeliverableDocuments/D5.2_v1.1.pdf`.

[160] B. Melander and M. Björkman. Trace-driven network path emulation. Technical report 2002-037, Department of Information Technology, Uppsala University, Sweden, 2002.

[161] M. Meyer, J. Sachs, and M. Holzke. Performance evaluation of a TCP proxy in WCDMA networks. *IEEE Wireless Communications*, 10(5):70–79, October 2003.

[162] Microsoft Corporation. Microsoft windows media player. `http://www.microsoft.com/windows/mediaplayer` Visited 2011-09-09.

[163] P. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), Nov. 1987.

[164] N. Moller, K. Johansson, and H. Hjalmarsson. Making retransmission delays in wireless links friendlier to TCP. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, volume 5, pages 5134–5139, Bahamas, Dec. 2004.

[165] B. Mukherjee and T. Brecht. Time-lined TCP for the TCP-friendly delivery of streaming media. In *Proceedings of the International Conference on Network Protocols (ICNP)*, pages 165–176, Osaka Japan, Nov. 2000.

[166] C. S. R. Murthy and B. S. Manoj. *Ad Hoc Wireless Networks*. Prentice Hall, 2004. ISBN 0-13-147023-X.

[167] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. RFC 2474 (Proposed Standard), Dec. 1998.

[168] NIST Internetworking Technology Group. NISTNet network emulation package. Available from `http://snad.ncsl.nist.gov/nistnet/`. Visited 2011-12-28.

[169] B. Noble, G. Nguyen, M. Satyanarayanan, and R. Katz. Mobile network tracing. RFC 2041 (Informational), Oct. 1996.

[170] B. Noble, M. Satyanarayanan, G. Nguyen, and R. Katz. Trace-based mobile network emulation. In *Proceedings of ACM SIGCOMM*, September 1997.

[171] S. Ostermann. Tcptrace. `http://www.tcptrace.org/` Visited 2011-09-09.

[172] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. *Proceedings of ACM SIGCOMM*, pages 303–314, Aug. 1998.

[173] V. Paxson and M. Allman. Computing TCP's retransmission timer. RFC 2988 (Proposed Standard), Nov. 2000.

[174] H. Persson, A. Brunstrom, and T. Ottosson. Utilizing cross-layer information to improve performance in JPEG2000 decoding. *Advances in Multimedia*, 2007, 2007.

[175] J. Postel. User datagram protocol. RFC 768 (Standard), Aug. 1980.

[176] J. Postel. Internet protocol. RFC 791 (Standard), Sept. 1981.

[177] J. Postel. Transmission control protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168, 6093.

[178] R. Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416 (Standard), Dec. 2002.

[179] J. G. Proakis. *Digital communications*. McGraw-Hill, Boston, 2000.

[180] J. Prokkola, P. Perala, M. Hanski, and E. Piri. 3G/HSPA performance in live networks from the end user perspective. In *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2009.

[181] S. Ramachandran and A. Jain. Web page stats: size and number of resources, 2010. Available at `http://code.google.com/speed/articles/web-metrics.html` Visited 2011-12-28.

[182] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001.

[183] K. Ratnam and I. Matta. WTCP: An efficient mechanism for improving wireless access to TCP services. *International Journal of Communication Systems*, 16:47–62, 2003.

[184] A. Ravichandran, M. Tacca, M. Welzl, and A. Fumagalli. LN-MAC: a cross-layer explicit loss notification solution for TCP over IEEE 802.11. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, Dec. 2008.

[185] RealNetworks Inc. Real player. `http://www.real.com` Visited 2011-09-09.

[186] F. Ren, X. Huang, F. Liu, and C. Lin. Improving TCP throughput over HSDPA networks. *IEEE Transactions on Wireless Communications*, 7(6):1993–1998, June 2008.

[187] J. Rendón, F. Casadevall, and D. Serarols. Snoop TCP performance over GPRS. *Proceedings of the IEEE Vehicular Technology Conference (VTC-Spring)*, May 2001.

[188] L. Rizzo. Dummynet: A simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, Jan. 1997.

[189] M. Rossi, L. Scaranari, and M. Zorzi. On the UMTS RLC parameters setting and their impact on higher layers performance. In *Proceedings of the IEEE Vehicular Technology Conference (VTC Fall)*, volume 3, pages 1827–1832, Orlando, Florida, USA, Oct. 2003.

[190] M. Sågfors, R. Ludwig, M. Meyer, and J. Peisa. Buffer management for rate-varying 3G wireless links supporting TCP traffic. In *Proceedings of IEEE Vehicular Technology Conference (VTC-spring)*, pages 675–679, Apr. 2003.

[191] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, Nov. 1984.

[192] K. Sandrasegaran, S. Reeves, H. Ramli, and R. Basukala. Analysis of hybrid ARQ in 3GPP LTE systems. In *Proceedings of the Asia-Pacific Conference on Communications (APCC)*, pages 418–423, Nov. 2010.

[193] M. J. Schervish. A review of multivariate analysis. *Statistical Science*, 2(4):396–413, 1987. Available at `http://projecteuclid.org/euclid.ss/1177013111`. Visited 2012-01-04.

[194] H. Schulzrinne and S. Casner. RTP profile for audio and video conferences with minimal control. RFC 3551 (Standard), July 2003.

[195] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. RFC 3550 (Standard), July 2003.

[196] M. Schwartz. *Mobile wireless communications*. Cambridge University Press, 2005.

[197] J. Shaikh, T. Minhas, P. Arlos, and M. Fiedler. Evaluation of delay performance of traffic shapers. In *Proceedings of the International Workshop on Security and Communication Networks (IWSCN)*, Karlstad, Sweden, May 2010.

[198] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27, Oct. 1948.

[199] A. Singh and S. Iyer. ATCP: Improving TCP performance over mobile wireless environments. In *Proceedings of the International Workshop on Mobile and Wireless Communications Network (MWCN)*, pages 239–243, Stockholm, Sweden, 2002.

[200] J. P. Singh, Y. Li, N. Bambos, A. Bahai, B. Xu, and G. Zimmermann. TCP performance dynamics and link-layer adaptation based optimization methods for wireless networks. *IEEE Transactions on Wireless Communications*, 6(5):1864–1879, 2007.

[201] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2):301–316, March 2002.

[202] R. Sperschneider, D. Homm, and L.-H. Chambat. Error resilient source coding with differential variable length codes and its application to MPEG Advanced Audio Coding. In *Proceedings of the Audio Engineering Society Convention 112*, 4 2002.

[203] W. Stallings. *Wireless Communications and Networks*. Prentice Hall, 2001.

[204] W. Stallings. *Data and computer communications*. Prentice Hall, 2007.

[205] M. Sternad. The wireless IP project. In *Proceedings of Radiovetenskap och Kommunikation*, Stockholm, Sweden, June 2002.

[206] M. Sternad. The SSF Wireless IP Project: Final Report, Oct. 2009. Submitted to the SSF. Available online at `http://www.signal.uu.se/Research/PCCWIP/WirelessIP_Finalreport.pdf` . Visited 2011-10-10.

[207] M. Sternad. Personal communication, Aug. 2011.

[208] M. Sternad. Publications and workshops within the Wireless IP project. Project web page, 2011. `http://www.signal.uu.se/Publications/wip.html`. Visited 2012-03-14.

[209] M. Sternad and D. Aronsson. Channel estimation and prediction for adaptive OFDM downlinks. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall)*, Orlando, Florida, USA, Oct. 2003.

[210] M. Sternad and S. Falahati. Maximizing throughput with adaptive M-QAM based on imperfect channel predictions. In *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Barcelona, Spain, Sept. 2004.

[211] M. Sternad, T. Ottoson, A. Ahlén, and A. Svensson. Attaining both coverage and high spectral efficiency with adaptive OFDM downlinks. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall)*, Orlando, Florida, USA, Oct. 2003.

[212] M. Sternad, T. Svensson, T.Ottosson, A. Ahlen, A. Svensson, and A. Brunstrom. Towards systems beyond 3G based on adaptive OFDMA transmission. In *Proceedings of the IEEE, Special Issue on Adaptive Transmission*, volume 95:12, pages 2432–2455, 2007.

[213] R. W. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison Wesley, Reading, 1994. ISBN 0-201-63346-9.

[214] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001 (Proposed Standard), Jan. 1997.

[215] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. Stream control transmission protocol (SCTP) partial reliability extension. RFC 3758 (Proposed Standard), May 2004.

[216] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol. RFC 2960 (Proposed Standard), Oct. 2000.

[217] T. Stivers, N. J. Enfield, P. Brown, C. Englert, M. Hayashi, T. Heinemann, G. Hoymann, F. Rossano, J. P. de Ruiter, K.-E. Yoon, and S. C. Levinson. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Science (PNAS)*, 106(26), June 2009.

[218] R. Susitaival, H. Wiemann, J. Östergaard, and A. Larmo. Internet access performance in LTE TDD. In *Proceedings of IEEE Vehicular Technology Conference (VTC-Spring)*, May 2010.

[219] A. Svensson. An introduction to adaptive QAM modulation schemes for known and predicted channels. *Proceedings of the IEEE*, 95(12), December 2007.

[220] A. Svensson, A. Ahlén, A. Brunstrom, T. Ottosson, and M. Sternad. An OFDM based system proposal for 4G downlinks. In *Proceedings of Multi-Carrier Spread Spectrum Workshop*, Oberpfaffenhofen, Germany, 2003.

[221] M. Taferner and E. Bonek. *Wireless Internet Access over GSM and UMTS.* Springer-Verlag, 2002.

[222] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A compound TCP approach for high-speed and long distance networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–12, Apr. 2006.

[223] A. S. Tanenbaum. *Computer Networks, 4th ed.* Prentice-Hall International, Inc., 2003. ISBN: 0-13-066102-3.

[224] Telecommunications Industry Association. *TIA/TSB-116-A: Voice quality recommendations for IP telephony.* TIA, 2006. Available at http://www.tiaonline.org/standards/technology/voip/documents/TSB116-Afinalforglobal.pdf.

[225] Telia Sonera. TeliaSonera first in the world with 4G services. Telia Sonera press release 2009-12-14, 2009. `http://www.teliasonera.com/en/newsroom/press-releases/2009/12/teliasonera-first-in-the-world-with-4g-services/` Visited 2012-03-14.

[226] The Bluetooth Special Interest Group. The official Bluetooth sig member website. `http://www.bluetooth.org/` Visited 2012-02-10.

[227] The Working Group for WLAN Standards. IEEE 802.11 wireless local area networks. `http://grouper.ieee.org/groups/802/11/` Visited 2011-09-02.

[228] J. Touch. Automating the initial window in TCP. Internet Draft (work in progress; expires 2012-07-17), July 2011. Available at `http://tools.ietf.org/html/draft-touch-tcpm-automatic-iw-02`. Accessed 2012-03-14.

[229] U.S. Naval Research Laboratory. Multi-generator MGEN, version 4, 2008. `http://cs.itd.nrl.navy.mil/work/mgen/` Visited 2012-03-14.

[230] F. Vacirca, A. D. Vendictis, A. Todini, and A. Baiocchi. On the effects of ARQ mechanisms on TCP performance in wireless environments. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, USA, Dec. 2003.

[231] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless. *Technical Report, Computer Science Dept., Texas A&M University*, February 1999.

[232] C. Vulkan and B. Heder. Congestion control in evolved HSPA systems. In *Proceedings of the IEEE Vehicular Technology Conference (VTC Spring)*, May 2011.

[233] L. Wang, G. Zhu, G. Su, and W. Wu. Improving TCP performance by TCP-Acknowledge agent based on AP ARQ over wireless links. *International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 710–713, 2007.

[234] L.-C. Wang and C.-H. Lee. A TCP-physical cross-layer congestion control mechanism for the multirate WCDMA system using explicit rate change notification. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA)*, pages 449–452, 2005.

[235] W. Wang, T. Ottosson, M. Sternad, A. Ahlén, and A. Svensson. Impact of multiuser diversity and channel variability on adaptive OFDM. In *Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall)*, Orlando, Florida, USA, Oct. 2003.

[236] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *Proceedings of SIGCOMM*, pages 374–385, Toronto, Ontario, Canada, 2011.

[237] WAP Forum Ltd. Wireless Application Protocol - architecture specification. *http://www.wapforum.org*, April 1998.

[238] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *Proceedings of the IEEE International Conference on Communications (ICC)*, Dresden, Germany, 2009.

[239] M. Welzl, M. Rossi, A. Fumagalli, and M. Tacca. TCP/IP over IEEE 802.11b WLAN: the challenge of harnessing known-corrupt data. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 280–284, Beijing, China, May 2008.

[240] A. Wennström, S. Alfredsson, and A. Brunstrom. TCP over wireless networks. Karlstad University Studies 2004:21, Karlstad University Press, 2004.

[241] World Wide Web Consortium. HTML 5: A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft, 2011. `http://www.w3.org/TR/html5/` Visited 2011-12-28.

[242] WWI. The Wireless World Initiative. Available online at `http://www.wireless-world-initiative.org/`. Visited 2011-10-10.

[243] WWRF. The Wireless World Research Forum. Available online at `http://www.wireless-world-research.org`. Visited 2011-10-10.

[244] WWRF. Wireless World Initiative New Radio. Available online at `http://www.ist-winner.org`. Visited 2011-10-10.

[245] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, Hong Kong, 2004.

[246] K. Yamamoto, H. Suzuki, N. Ishikawa, M. Miyake, and H. Inamura. A TCP flow control scheme for 3G mobile communication networks. In *Proccedings of the International Conference on Computer Communications and Networks (ICCCN)*, pages 229–236, Oct. 2006.

[247] P. Yang, W. Luo, L. Xu, J. Deogun, and Y. Lu. TCP congestion avoidance algorithm identification. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS)*, pages 310–321, June 2011.

[248] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA)*, Santa Cruz, California, USA, Dec. 1994.

[249] I. Yeom. ENDE: An end-to-end network delay emulator. Master's thesis, Texas A&M University, College Station, Texas, USA, 1998.

[250] C. Zhang and V. Tsaoussidis. TCP Real: Improving real-time capabilities of TCP over heterogeneous networks. *Proceedings of the The 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2001.

Acronyms and abbreviations

3GPP Third Generation Partnership Project, page 33

3GPP LTE 3GPP Long Term Evolution, page 33

3GPP LTE Advanced 3GPP Long Term Evolution - Advanced, page 33

ACK TCP ACKnowledgement, page 38

ADSL Asymmetric Digital Subscriber Line, page 10

AIMD Additive Increase, Multiplicative Decrease, page 42

AMPS Advanced Mobile Phone System, page 29

ARIB The Association of Radio Industries and Businesses, Japan, page 33

ARQ Automatic Repeat Request, page 38

ATIS The Alliance for Telecommunications Industry Solutions, USA, page 33

ATM Asynchronous Transfer Mode, page 155

AWGN Additive White Gaussian Noise, page 118

BER Bit Error Rate, page 87

bin A slot in time and frequency consisting of 120 symbols with a duration of 0.667 ms, page 95

BMP Bit MaPped image format, page 76

BPSK Binary Phase-Shift Keying, page 118

CCSA China Communications Standards Association, page 33
CDMA Code Division Multiple Access, page 23
CDO Corruption Detection Option, page 53
CDPD Cellular Digital Packet Data, page 29
CNO Corruption Notification Option, page 53
CoMP Cooperative MultiPoint transmission, page 172
CRC Cyclic redundancy check, page 20
CSMA Carrier Sense Multiple Access, page 21
CTCP Compound TCP, page 45
CWND TCP Congestion WiNDow, page 40
D-AMPS Digital Advanced Mobile Phone System, page 29
DCCP Datagram Congestion Control Protocol, page 47
DNS Domain Name System, page 38
ECN Explicit Congestion Notification, page 62
EDGE Enhanced Data rates for GSM Evolution, page 29
ELN Explicit Loss Notification, page 62
ENDE End-to-end Network Delay Emulator, page 70
ETSI European Telecommunications Standards Institute, page 33
EV-DO Evolution-Data Optimized, page 65
FDMA Frequency Division Multiple Access, page 21
FEC Forward Error Correction, page 167
FIN TCP FINish flag, page 38
GPRS General Packet Radio Service, page 29
HSDPA High Speed Downlink Packet Access, page 33
HSPA High Speed Packet Access (HSDPA and HSUPA), page 33
HSUPA High Speed Uplink Packet Access, page 33
I-TCP Indirect TCP, page 64
IETF Internet Engineering Task Force, page 11
IMT International Mobile Telecommunications, page 32

ISO International Organization for Standardization, page 11
ITU International Telecommunication Union, page 30
kB kilobyte (1024 bytes), page 139
kbit kilobit (1000 bits), page 139
LN-MAC Loss-notifying MAC, page 54
LOS Line-Of-Sight, page 18
MAP Maximum A Posteriori probability estimate, page 76
MIMO Multiple Input Multiple Output, page 19
MOS Mean Opinion Score, page 158
MSDU MAC Service Data Unit, page 45
MSR Mobile Support Router, page 64
MTU Maximum Transmission Unit, page 45
NLOS Non-Line-Of-Sight, page 18
NMSE Normalized Mean Squared Error, page 97
NMT-450 Nordic Mobile Telephony (450 MHz), page 29
ns Network Simulator, page 70
OFDM Orthogonal Frequency-Division Multiplexing, page 24
OFDMA Orthogonal Frequency Division Multiple Access, page 24
ONE Ohio Network Emulator, page 70
OSI ISO Open Systems Interconnection, page 11
PAN Personal Area Network, page 27
PCM Pulse Coded Modulation, page 76
PESQ Perceptual Evaluation of Speech Quality, page 158
PR-SCTP Partially Reliable SCTP, page 76
PRTP Partially Reliable Transport Protocol, page 76
PSH TCP PuSH flag, page 38
QAM Quadrature Amplitude Modulation, page 96
RLC Reliable Link Control, page 58
RST TCP ReSeT flag, page 38

RTO TCP Retransmission TimeOut, page 39
RTT Round-Trip Time, page 39
RWND TCP Receiver WiNDow, page 40
SACK TCP Aelective ACKnowledgement, page 43
SCTP Stream Control Transmission Protocol, page 47
SDMA Space-Division Multiple Access, page 19
SINR Signal to Interferer and Noise Ratio, page 96
SNMP Simple Network Monitoring Protocol, page 38
SNR Signal to Noise Ratio, page 96
SSID Service Set ID, page 27
SSTHRES TCP Slow Start THREShold, page 40
SYN TCP SYNchronize flag, page 38
TCP-L TCP-Lite, page 77
TCP-RC TCP Receiver-Centered, page 76
TDMA Time Division Multiple Access, page 22
TL-TCP Time-Lined TCP, page 76
TTA Telecommunications Technology Association, Korea, page 33
TTC Telecommunication Technology Committee, Japan, page 33
UMTS Universal Mobile Telecommunications System, page 29
URG TCP URGent flag, page 38
VoIP Voice over IP, page 155
WCDMA Wideband CDMA, page 67
WINNER Wireless world INitiative NEw Radio, page 32
WIP Wireless IP (project), page 94
WIPEMU Wireless IP EMUlator, page 94
WLAN Wireless Local Area Network, page 27
WWAN Wireless Wide Area Network, page 28
WWI Wireless World Initiative, page 32
WWRF Wireless World Research Forum, page 32

A Cross-Layer Perspective on Transport Protocol Performance in Wireless Networks

Communication by wireless technologies has seen a tremendous growth in the last decades. Mobile phone technology and wireless broadband solutions are rapidly replacing the last-hop wireline connectivity for telephones and Internet access. Research has, however, shown that Internet traffic can experience a performance degradation over wireless compared to wired networks. The inherent properties of radio communication lead to a higher degree of unreliability, compared to communication by wire or fiber. This can result in an increased amount of transmission errors, packet loss, delay and delay variations, which in turn affect the performance of the main Internet transport protocols TCP and UDP.

This dissertation examines the cross-layer relationship between wireless transmission and the resulting performance on the transport layer. To this end, experimental evaluations of TCP and UDP over a wireless 4G downlink system proposal are performed. The experiment results show, in a holistic scenario, that link-level adaptive modulation, channel prediction, fast persistent link retransmissions, and channel scheduling, enables the transport protocols TCP and UDP to perform well and utilize the wireless link efficiently. Further, a novel approach is proposed where a modified TCP receiver can choose to accept packets that are corrupted by bit errors. Results from network emulation experiments indicate that by accepting and acknowledging even small amounts of corrupted data, a much higher throughput can be maintained compared to standard TCP.

ISBN 978-91-7063-411-6

ISSN 1403-8099

DISSERTATION | Karlstad University Studies | 2012:7