Automatic control in TCP over wireless

NIELS MÖLLER

Licentiate Thesis
Stockholm, Sweden, 2005

TRITA-S3-REG-0504
ISSN-1404-2150
ISBN-91-7178-143-9

KTH
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av licentiatexamen i telekommunikation 2005-09-30 i D34.

Tryck: Universitetsservice US AB

Abstract

Over the last decade, both the Internet and mobile telephony have become parts of daily life, changing the ways in which we communicate and search for information. These two distinct technologies are now slowly merging. The topic of this thesis is TCP over wireless, and the automatic control that is used within the system, from the link-layer power control to the end-to-end congestion control. The thesis consists of three main contributions.

The first contribution is a proposed split-connection scheme for downloads to a mobile terminal. A wireless mobile terminal requests a file or a web page from a proxy, which in turn requests the data from a server on the Internet. During the file transfer, the radio network controller (RNC) sends radio network feedback (RNF) messages to the proxy. These messages include information about bandwidth changes over the radio channel, and the current RNC queue length. A novel control mechanism in the proxy uses this information to adjust the sending rate. The stability and convergence speed of the proxy controller is analyzed theoretically. The performance of the proposed controller is compared to end-to-end TCP Reno, using `ns-2` simulations for some realistic scenarios. It is shown that the proxy controller is able to reduce the response time experienced by users, and increase the utilization of the radio channel. The changes are localized to the RNC and the proxy; no changes are required to the TCP implementation in the terminal or the server.

The second contribution is the analysis of an uplink channel using power control and link-layer retransmissions. To be able to design the link-layer mechanisms in a systematic way, good models for the link-layer processes, and their interaction with TCP, are essential. The use of link-layer retransmissions transforms a link with constant delay and random losses into a link with random delay and almost no losses. As seen from the TCP endpoints, the difference between such a link and a wired one is no longer the loss rate, but the packet delay distribution. Models for the power control and link-layer retransmissions on the link are used to derive the packet delay distribution, and its impact on TCP performance is investigated.

The final contribution involves ways to optimize the link-layer processes. The main result is that TCP performance, over a wireless link with random retransmission delays, can be improved by adding carefully chosen artificial delays to certain packets. The artificial delays are optimized off-line and applied on-line. The additional delay that is applied to a packet depends only on the retransmission delay experienced by that same packet, and this information is available locally at the link.

Preface

Nätverk, *nätförband (Lat.* opus reticulātum*),* bygnk., *ett murförband, hvari stenarna äro ställda på hörn, så att fogarna bli diagonala, (se fig.), stundom användt i den gamla romerska och någon gång i den fornkristna byggnadskonsten. Äfven en nätliknande ornering kallas nätverk. Jfr Mur, sp. 1372.*

G. H. W Upmark, Nordisk familjebok, 1914.

This thesis is the result of research in the area where computer networking and the theory of automatic control intersect. The project "Towards a self-regulating Internet" at KTH was started in 2002. This project is a collaboration between the automatic control group at the department of Signals, Sensors, and Systems at KTH, and the communications networking group, which at the time[1] was part of the department of Microelectronics and Information Technology at KTH. I was the first graduate student in automatic control, to be involved in this project.

In recent years, applying control theory to computer networking, as an emerging research field, has attracted interest from three different communities and cultures: the control theory community, the networking community, and the telecommunication community.

When I got involved in the project, I was well versed in mathematics and computer science, and I knew the basics of control theory. I had considerable practical experience with computer networking, from the period I spent helping building the first dorm network in Linköping, and from programming various network applications, but I was not familiar with networking theories and research. The traditional telecom world was, and still is, to some degree, an alien world.

So it has been interesting and rewarding to try to get the different traditions and viewpoints to fit together. For a slightly exaggerated example, say that a mobile and a base station exchange packets, and that packets that are lost due to noise on the radio are retransmitted (this is called link-layer retransmissions, and is one central issue in this thesis). A control theorist would think about the base station and mobile as a feedback control system, and draw a simplified block diagram. A computer network researcher might view the retransmissions as a dubious hack,

[1]Today, the organization is different. Both groups are now part of the same department, and have the fortune of being located in the same building.

intended to make the important end-to-end protocol TCP work better, and then implement it in the `ns-2` network simulator. A telecommunications researcher might view the retransmissions as a quality of service parameter, which the operator could charge extra for. Since the work has been done within the automatic control group, I have tried to primarily keep a control perspective sometimes translating networking concepts into the language of automatic control.

The research described in this thesis has been partially funded by the Swedish Research Council, by the Swedish Foundation for Strategic Research, by the European Commission through the HYCON and RUNES projects, and by the Swedish Center for Internet Technologies.

Photos are kindly provided by Kjell Enblom and Todd Klassy. The latter photo is distributed under the Creative Commons attribution license.

I would like to thank my advisor Karl Henrik Johansson, who has read and commented on my draft papers more times than anybody else, and Carlo Fischione, who has helped me to get the radio communications issues right. For the radio network feedback work, thanks goes to Inés Cabrera Molero, who performed the simulations and wrote the needed `ns-2` code, as part of an excellent master of science thesis, and also to the people at Ericsson who were involved: Åke Arvidsson, Justus Petersson, and Robert Skog.

I would like to thank my fellow PhD students and the other friendly people in the control group. Finally, I would like to thank Cecilia for her encouragement, ever since I first started on this project.

Contents

Chapter 1

Introduction

The main theme of this thesis is the use of TCP/IP *over wireless links, and interactions between processes in different layers.*

1.1 Motivation

Over the last decade, both the Internet and mobile telephony have become parts of daily life, changing the ways we communicate and search for information. These two distinct tools are now slowly merging, both at the surface, and in the underlying communication infrastructure.

A wireless mobile Internet means that mobile gadgets are first-class citizens of the Internet. Any communication task, e.g., email, web browsing, Internet radio, or peer-to-peer file sharing, that is possible with a stationary computer connected to the Internet, should be equally possible with a suitable mobile gadget.

Enabling a wireless mobile Internet is a huge task. One prerequisite is that TCP/IP, the two most important protocols on the Internet, must work satisfactorily across a heterogeneous network consisting of an assortment of stationary and mobile devices, connected by different types of wired and wireless links.

The focus of this thesis is on the TCP protocol, the problems encountered when using TCP over cellular radio links, and the various feedback control loops used within the system. This first chapter provides an introduction to the TCP-over-wireless problem, and outlines the contributions in this thesis. Chapter 2 provides additional background material and references. The main contributions are found in Chapters 3–5. Finally, Chapter 6 discusses the results, and outlines some directions for future research.

1.2 IP networking and the end-to-end principle

Layered design of communication systems is a modularization technique, where each layer at a particular node needs to know how to communicate with the layers

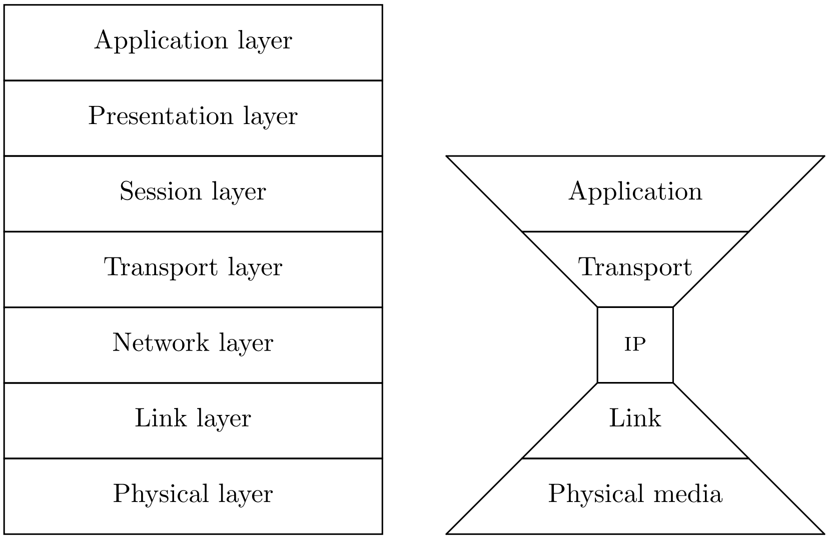

Figure 1.1: Left: The OSI networking stack. Right: The IP networking model.

directly above and below at the *local* node, but only to the *same* layer at remote nodes. E.g., the Internet Protocol (IP) layer (or networking layer) needs to know how to transmit IP packets using the local link-layer, but it does not need to know anything about the receiver's link-layer.

The Open Systems Interconnection (OSI) reference model specifies seven layers, illustrated to the left in Figure 1.1. From bottom to top: physical layer, link layer, networking layer, transport layer, session layer, presentation layer and application layer. IP networks uses a somewhat simpler model. At the core, we have the IP layer, corresponding to the networking layer of the OSI model. The IP layer is a fairly primitive packet transport service, which provides unreliable best effort packet delivery between nodes, identified by their IP addresses (32 bits for IP version 4, 128 bits for IP version 6). Packets may be dropped, duplicated or delivered out of order.

The power of IP, the Inter-network protocol, is that it is used to communicate across heterogeneous networking environments, e.g., ethernet, point-to-point links, and cellular networks. These different technologies are accommodated as the abstract notion of a link and a link layer. The link layer is directly below the IP layer in the networking stack, and it provides IP packet transport between nodes that share the same link. There are multitude of different link types in the Internet, and a multitude of transport layer and application layer protocols, but they are all used with a single packet transport service: IP. This is illustrated to the right in Figure 1.1.

Above the IP layer, we have the transport layer, where the Transmission Control

Protocol (TCP) is the protocol of primary interest. TCP is responsible for dividing a data stream into packets, ensure reliable delivery even when the IP layer loses, reorders, or duplicates packets, and at the same time it senses the state of the network to avoid overloading it. In the context of IP networking (as opposed to the construction of applications and application protocols), everything above the transport layer is usually referred to as "application layer", with no subdivision into session layer, presentation layer, etc.

There are no sophisticated mechanisms for resource reservation or allocation built in to the IP layer; the normal response for a router or link that is overloaded is to simply discard the packets it cannot handle, and leave to the communicating endpoints to sort things out best they can. This is an important design choice: The network core is simple, while endpoints must be quite sophisticated in order to work well together with the network. This design is known as the *end-to-end principle.*

1.3 The TCP protocol

The dominating transport layer protocol is TCP. It is used for all kinds of data streams: long-lived low bandwidth interactive traffic, e.g., telnet and ssh sessions, short-lived file transfers, e.g., web traffic, longer lived bulk transfers, e.g., ftp and file sharing, and almost real-time traffic, e.g., Internet radio.

A TCP connection is a bidirectional, flow controlled, reliable stream of data between two endpoints, identified by IP-address and port number. Our primary interest is in TCP connections for transfer of smaller or larger files. For this TCP usage, it is desirable to get as much data as possible through the network, while at the same time we must avoid overloading the network, and share available bandwidth in a fair way with other users.

TCP uses a sliding window flow control. The window limits the amount of data that can be sent without waiting for acknowledgement (ACK) from the receiver. When the window is constant, this results in the so called "ACK clock"; the timing of each sent packet is determined by the reception of the ACK for an earlier packet.

One can think about the sliding window and the ACK clock as a peculiar inner control loop which determines the sending rate; when the roundtrip time fluctuates, the sliding window gives an average sending rate of one full window per average roundtrip time. The window size is adjusted depending on received ACKs, and it is the details of this outer loop that differ between TCP variants.

The objective of the TCP window control is to get a high throughput, close to the connection's fair share of the available bandwidth, and at the same time avoid overloading the network. The fair share can vary due to varying amounts of competing cross traffic, and also due to network changes such as to routing updates or radio links with time-varying capacity.

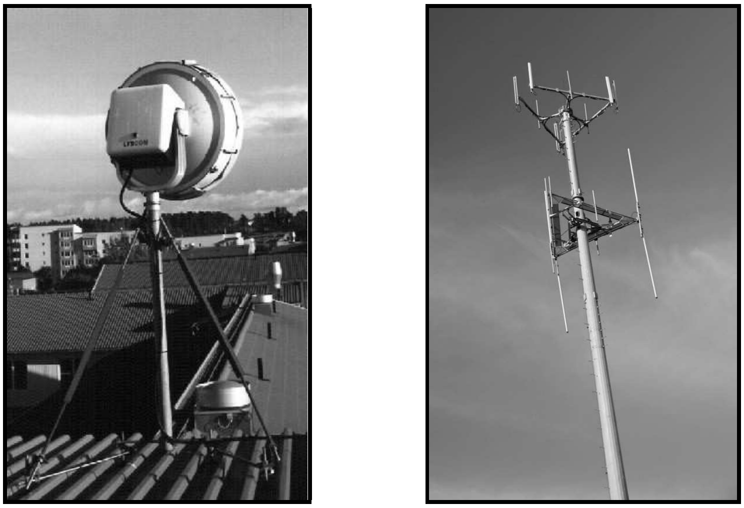

Figure 1.2: Left: Point-to-point microwave link, which connected the student dorm network in Linköping to the the campus network and the Internet, 1993–1996. Photo by Kjell Enblom, RydNet. Right: Cell phone tower in Oregon. Photo by Todd Klassy.

1.4 Wireless links

Until the 1990s, the Internet consisted almost exclusively of wired links, consisting of copper cables and optical fibers. The occasional point-to-point microwave links were usually quite reliable high power devices using parabolic antennas, such as the link at the left of Figure 1.2.

Today, mobile wireless communication is common place, thanks to the infrastructure of mobile telephony. Radio communication in the cellular telephony system is more difficult than a point-to-point microwave link, due to lower transmission power, mobility, and often lack of a direct line-of-sight between transmitter and receiver. The mobile telephony system is tailored to handle these difficulties of the available radio channel.

For the future, we need to integrate the mobile telephony system with the Internet infrastructure. To make the mobile terminal a first class citizen of the Internet, we need to use TCP and IP all the way out to the terminal. To make such an integrated system work efficiently is challenging, because the wireless link available to the terminal has different characteristics than the wired links that TCP was originally designed for.

The operation of a wireless link is much more complex than traditional electrical or optical links. The various mechanisms used, and the corresponding input and output signals, are shown in Figure 1.3.

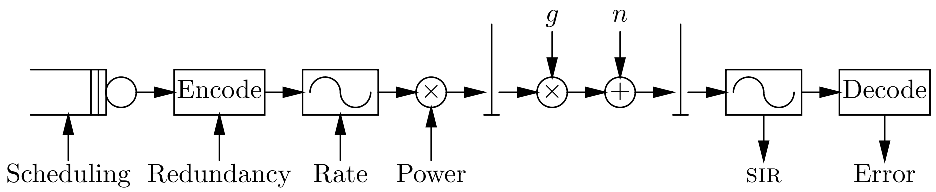

Figure 1.3: Wireless link mechanisms. From left to right: Buffer of frames to send, channel coder, modulator, transmission amplifier, transmission antenna, channel gain, channel noise, reception antenna, demodulator, and channel decoder.

On the sending side, at the left in the figure, we have a buffer of frames to send, controlled by a scheduler which decides which frame to transmit in which time slot. Next, a channel coder, adding redundancy for error correction coding, and a modulator, selecting the modulation scheme and the particular waveform to use. Finally, a power amplifier and the transmission antenna.

The disturbances on the radio channel can usually be accurately modelled as a multiplicative gain disturbance and an additive noise, g and n in the figure. These are time-varying due to movement of antennas and obstacles, and due to varying level of background noise and interference.

On the right, we have the receiver, consisting of a demodulator that detects the used waveform, and a decoder that tries to correct symbol errors. Besides the actual data, the output signals from the receiver are the signal to interference ratio (SIR)[1] and a per frame indication saying if the frame could be recovered after error correction. The available control signals are the transmission power, the sending rate used by the modulator, the amount of redundancy added by the channel coder, and the scheduling, i.e., decisions on which frame to transmit when, as well as possibly deciding to stay silent when conditions are too bad.

The figure shows a single wireless link. In practice, we have multiple links operating in the same area and frequency range. In this thesis, the focus is on cellular architecture, where base stations are distributed geographically, each base station handles a number of mobile terminals, and the access scheme used is wideband code division multiple access (WCDMA).

In a cellular system, the design tradeoffs are quite different for the uplink, i.e., transmission from a mobile terminal to the base station, and the downlink, i.e., transmissions from the base station to the mobile terminals. In the uplink, capacity is limited by the interference between mobile terminals, which makes power control crucial. The performance of the power control influences not only the battery life of the terminals, but also the overall capacity and stability of the system. The power used by one terminal appears as an additive disturbance for the other users. In

[1]The SIR compares the power of the signal to the power of the interference from other users. This is different from the signal to noise ratio, which compares the power of the signal to the power of the thermal noise. In cellular systems, interference usually dominates over thermal noise.

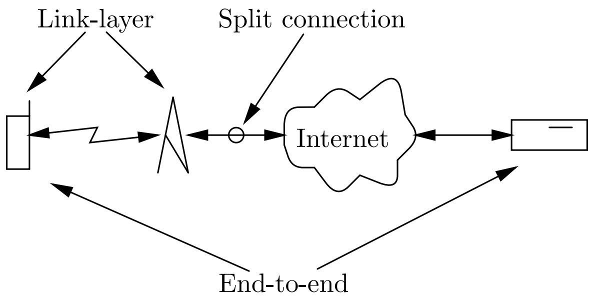

Figure 1.4: Approaches to the TCP over wireless problem.

the downlink, on the other hand, the base station can use scheduling or orthogonal modulation to limit the interference between transmissions to individual terminals. Instead, the downlink capacity is limited by the transmission power of the base station, and by interference from neighboring base stations.

The various transmission parameters, e.g., rate, redundancy, scheduling and power, are controlled using cascaded feedback loops, working on different timescales. Power control based on the received SIR is the fastest loop, working on a 1 ms time scale, scheduling works on a timescale of time slots, 10 ms, while adaptation of the rate and of the amount of redundancy is typically another order of magnitude slower.

1.5 TCP over wireless

When sending IP packets over a wireless link, the IP layer sees a link where one or more of the capacity, the loss rate, and the delay characteristics varies with time. Use of the available link-layer control mechanisms (power control, rate adaptation, forward error correction, and retransmission scheduling) enables us to make trade-offs between capacity, loss and delay, but it is not possible to achieve constant high capacity, low delay, and low loss, which is characteristic of a wired link.

The TCP congestion control mechanisms were designed to adapt to the type of disturbances that are common in a wired network, where available bandwidth varies due to cross traffic and occasional routing or capacity changes, and delays are caused by constant propagation delays and queueing delays.

When TCP is used over the cellular infrastructure, the result is often that both end-to-end throughput and radio link utilization are quite poor. This is because the dynamic properties of TCP and of wireless links do not fit well together.

The objective of work on the TCP-over-wireless problem is to achieve both good end-to-end throughput and efficient utilization of the radio resources, preferably with as small changes to existing infrastructure and protocols as possible. The wild fauna of proposed solutions to the TCP-over-wireless problem can be classified as follows:

End-to-end: Improve the TCP-protocol to adapt better to the disturbances from wireless links.

Split-connection: Introduce a proxy in the wired network, close to the wireless link. The proxy acts as an endpoint for a TCP connection over the wired network, and communicates with the wireless terminal using a specialized TCP version over the wireless part of the network, or even using some completely different protocol.

Link-layer: Design link-layer tradeoffs that results in link properties that plain TCP handles better.

These approaches are illustrated in Figure 1.4.

The proxy mechanism described in Chapter 3 falls into the split-connection class. It uses an explicit HTTP proxy. Another variant of the split-connection approach (which is not discussed in detail in this thesis) is called "performance enhancing proxies". Such a proxy does not act as a TCP endpoint, instead, it can insert, delay or delete packets in the stream, usually acknowledgements from the mobile terminal [7]. By doing so, the proxy can hide some of the wireless disturbances from the endpoint on the wired network, or force the end point to react differently to events on the wireless link.

Chapter 5 follows the link-layer approach. It describes a way to change the packet delay distribution by adding some additional delay, which reduces the probability of spurious timeout and improves TCP throughput.

It is likely that mechanisms from more than one of these classes should be used. An important aspect of the problem is to understand how the needed disturbance rejection work should be divided between end-to-end mechanisms and mechanisms closer to the wireless link.

1.6 Contributions

This thesis contains three fairly independent contributions.

- Chapter 3 proposes a split-connection proxy solution for TCP over the high-speed downlink packet access (HSDPA) channel in WCDMA. The connection between the proxy and the terminal does not use standard TCP congestion control. Instead the proxy uses feedback control and feedforward control, based on explicit messaging from the radio network controller (RNC), which has detailed knowledge of the current radio conditions. Simulations and quantitative results have been presented as

  I. Cabrera Molero, N. Möller, J. Petersson, R. Skog, Å. Arvidsson, O. Flärdh, and K. H. Johansson. Cross-layer adaptation for TCP-based applications in WCDMA systems. In *IST Mobile & Wireless Communications Summit*, Dresden, 2005.

The stability analysis will be presented as

> N. Möller, I. Cabrera Molero, K. H. Johansson, J. Petersson, R. Skog, and Å. Arvidsson. Using radio network feedback to improve TCP performance over cellular networks. In *IEEE CDC-ECC*, Seville, 2005.

The `ns-2` simulations for this chapter were done by Inés Cabrera Molero, as part of her Master Thesis project [33].

- Chapter 4 considers the uplink in a cellular system with power control and local retransmissions. Models for all layers from the power control up to TCP allow quantitative prediction of TCP performance degradation. This work is the chronologically first, and has been presented as

  > N. Möller and K. H. Johansson. Influence of power control and link-level retransmissions on wireless TCP. In *Quality of Future Internet Services*, volume 2811 of *Lecture Notes in Computer Science.* Springer-Verlag, Stockholm, 2003.

  > K. Jacobsson, N. Möller, K. H. Johansson, and H. Hjalmarsson. Some Modeling and Estimation Issues in Control of Heterogeneous Networks. In *Mathematical Theory of Networks and Systems*, Leuven, 2004.

  > N. Möller, C. Fischione, K. H. Johansson, F. Santucci, and F. Graziosi. Modeling and control of IP transport in cellular radio links. In IFAC *World Congress*, Prague, 2005.

- Chapter 5 describes a way to make a wireless link with a discrete delay distribution more friendly to TCP, by adding artificial delays to certain packets. This work has been presented as

  > N. Möller, K. H. Johansson, and H. Hjalmarsson. Making retransmission delays in wireless links friendlier to TCP. In *IEEE CDC*, Bahamas, 2004.

  and also at MTNS 2004 above.

Chapter 2

Background

TCP *over wireless is a complex system with several cascaded feedback loops. This chapter describes the use of automatic control in each subsystem.*

2.1 Congestion control in TCP

The objective of congestion control is to keep the load of the network close to the available capacity, and at the same time share the available capacity fairly between flows. Fairness is a peripheral issue in this thesis; but it must be noted that fairness is an important constraint in the design of transport protocols for the Internet.

The TCP protocol was developed in the late 1970s, resulting in the Internet Standard RFC 793 [15]. The principles for TCP congestion control were developed a few years later, in response to experience of "congestion collapses" in the Internet [24].

Window-based control

The most important concept in TCP congestion control is that of the *congestion window*. The window is the amount of data that has been sent, but for which no acknowledgement has yet been received. A constant congestion window means that one new packet is transmitted for each ACK that is received.

The sending rate is controlled indirectly by adjusting the congestion size. The standard way of doing this is documented in RFC 2581 [2], usually referred to as TCP Reno. It is described in this section. Two other common variants are TCP NewReno [18] and TCP with selective acknowledgements (SACK) [31, 19]. Before explaining the control mechanisms, we have to look into how TCP detects packet losses.

Acknowledgements and loss detection

At the receiving end, acknowledgement packets are sent in response to received data packets. TCP uses cumulative acknowledgements: Each acknowledgement includes a sequence number that says that *all* packets up to that one has been received. Equivalently, the acknowledgement identifies the next packet that the receiver expects to see.

When packets are received out of order, each received packet results in an acknowledgement, but they will identify the largest sequence number such that all packets up to that number has been received. E.g., if packets 1, 2, 4, and 5 are received, four acknowledgements are generated. The first says "I got packet #1, I expect packet #2 next", while the next three acknowledgements all say "I got packet #2, I expect packet #3 next". The last two acknowledgements are *duplicate* ACKs, since they are identical to some earlier ACK.

On the sending side, there are several possible reasons why duplicate ACKs are received: Packets delivered by the network out of order, packets dropped by the network, and ACK packets duplicated by the network.

Packet losses are detected by the sender in two ways:

- Timeout. If a packet is transmitted and no ACK for that packet is received within the retransmission timeout interval (RTO), the packet is considered lost.

- Fast retransmit. If three duplicate ACKs are received, the "next expected packet" from these ACKs is considered lost. Note that this can not happen if the congestion window is smaller than four packets.

Packets that are lost, as detected by either of these mechanisms, are retransmitted. Furthermore, congestion control actions are also based on these loss signals, as described below.

The value for RTO is not constant, but based on measured average and variation of the RTT. It is also modified by the exponential backoff mechanism.

TCP congestion control state

There are four distinctive states in the TCP congestion control, illustrated in Figure 2.1, and two state variables related to congestion control: The congestion window *cwnd* and the slow start threshold *ssthresh*. Typical initial values when TCP leaves the idle state and enters the slow start state are a *cwnd* of 2 packets, and a *ssthresh* that is the maximum value allowed by the wire protocol and by the receiving end.

We look at the operation of each of the four states in turn.

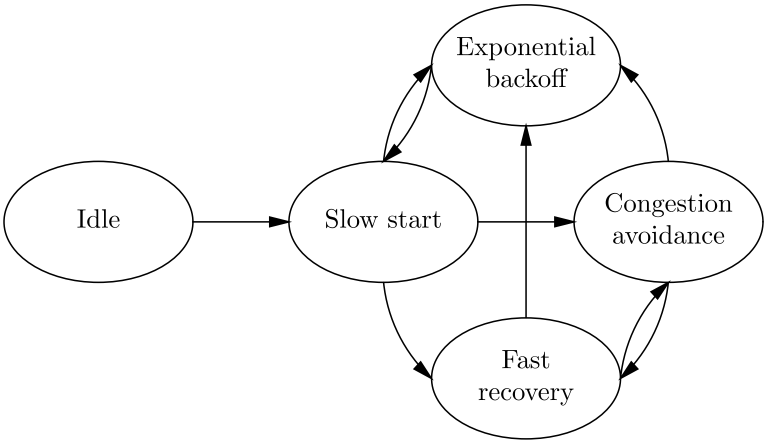

Figure 2.1: TCP state diagram. Transitions back to the idle state are omitted.

Slow start

The slow start state is the first state entered when a flow is created, or when a flow is reactivated after being idle. The slow start state can also be entered as the result of a timeout. In this state, *cwnd* is increased by one packet for each non-duplicate ACK. The effect is that for each received ACK, *two* new packets are transmitted. This implies that the congestion window, and also the sending rate, increases exponentially, doubling once per RTT.

It may seem strange to refer to an exponential increase of the sending rate as "slow start"; the reason is that in the early days, TCP used a large window from the start, and the introduction of the slow start mechanism did slow down connection startup.

Slow start continues until either

- $cwnd > ssthresh$, in which case TCP enters the congestion avoidance state, or
- a timeout occurs, in which case TCP enters the exponential backoff state, or
- three duplicate ACKs are received, in which case TCP enters the fast recovery state.

The motivation for the slow start state is that when a new flow enters the network, and there is a bottleneck link along the path, then the old flows sharing that link need some time to react and slow down before there is room for the new flow to send at full speed.

Congestion avoidance

In congestion avoidance mode, *cwnd* is increased by one packet per RTT (if *cwnd* reaches the maximum value, it stays there). This corresponds to a linear increase in the sending rate. On timeout, TCP enters the exponential backoff state, and on three duplicate ACKs, it enters the fast recovery state.

The motivation for this congestion avoidance mechanism is that since TCP does not know the available capacity, it has to probe the network to see at how high a rate data can get through. Aggressive probing would make the system unstable, and a single packet increase seems to work well in practice.

Exponential backoff

TCP enters the exponential backoff mode after timeout. Several actions are taken when entering this state:

- The lost packet is transmitted.
- The state variables are updated by $ssthresh \leftarrow cwnd/2$, $cwnd \leftarrow 1$ packet.
- The RTO value is doubled.

If the retransmission timer expires again with no ACK for the retransmitted packet, the packet is repeatedly retransmitted, RTO is doubled, and *ssthresh* is set to 1 packet [16]. The upper bound for the RTO is on the order of one or a few minutes.

Exponential backoff continues until an acknowledgement for the packet is received, in which case TCP enters the slow start phase, or the TCP stack or application gives up and closes the connection.

The motivation for the exponential backoff mechanism is that timeouts, in particular repeated timeouts, are a sign of severe network congestion. In order to avoid congestion collapse, the load on the network must be decreased considerably and repeatedly, until it reaches a level with a reasonably small packet loss probability.

Fast recovery

TCP enters the fast recovery state after it detects three duplicate ACKs. When entering this mode, the first actions of TCP is to retransmit the lost packet, and set $ssthresh \leftarrow cwnd/2$.

TCP then continues to send new data at approximately the same rate, one new packet of data for each received duplicate ACK. In RFC 2581 [2], this is described using a fairly complex procedure that artificially inflates *cwnd*.

If no ACK for the retransmitted packet is received within the RTO interval, TCP enters the exponential backoff state. Otherwise, when an ACK for the retransmitted packet is finally received, TCP sets $cwnd = ssthresh$, i.e., half the *cwnd* value at the start of the recovery procedure, and enters the congestion avoidance state.

If more than one packet is lost within the same window, fast recovery is limited in that it can recover only one packet per RTT. This is the main problem addressed by TCP NewReno and TCP SACK.

The motivation for the fast recovery mechanism is that the reception of duplicate ACKs indicates that the network is able to deliver new data to the receiver. Hence, the network is not severely congested, and we can keep inserting new packets into the network at the same rate as packets are delivered, at least for a while.

On the other hand, the loss of a packet also indicates that the network is on the border of congestion. At the end of the fast recovery procedure, *cwnd* is halved. TCP restarts the probing of the congestion avoidance state at a lower sending rate, at which it did not get any losses.

It should also be noted that halving the *cwnd* also implies that TCP will stay silent for about half an RTT, waiting for ACKs that reduce the number of outstanding packets, until the outstanding packets match the new window size.

Control theoretic view of congestion control

In control theoretic terms, a mechanism for congestion control of a flow in the network must somehow estimate the flow's fair share of the available bandwidth, and then use a sending rate based on this estimate. The main difficulties in doing this is that:

- Information about the network state, available at the endpoints, is scarce.
- The value being estimated, the fair share of available capacity, is time-varying and subject to disturbances.
- Sending rates based on an overestimation of the true value will quickly lead to queues building up in the network, leading to both packet losses and longer roundtrip times for those packets that are not lost.

A router in the network can be expected to know the capacities of attached links, the arrival rate for recent traffic, and the lengths of its queues. But, in the spirit of the end-to-end principle, routers should not be expected or required to maintain any per flow information. The signalling from the network infrastructure to TCP end nodes is severely limited. The signals available to end nodes are the arrival and timing of ACKs, and, if supported by the network, a single bit of explicit congestion notification (ECN) attached to each ACK [37].

Disturbance sources include varying levels of TCP cross traffic, varying levels of non-congestion controlled cross traffic, routing and topology changes, and, for wireless links, capacity changes due to variations of the radio channel.

The ACK clock

When TCP's congestion window is kept constant, the sender transmits one new packet for each received ACK, which is referred to as the ACK clock. The number of packets inside the network (be they data packets or ACKs) is kept constant. The ACK clock is based on the idea that by controlling the number of packets inside the network, we can control the load of the network.

> *... the packet flow is what a physicist would call 'conservative': A new packet isn't put into the network until an old packet leaves. The physics of flow predicts that systems with this property should be robust in the face of congestion.*
>
> Van Jacobson, [24]

The average transmission rate is one window of data per roundtrip time. In TCP congestion control, the control signal is the window size, and the actual sending rate is controlled only indirectly by adjusting the window.

Additive increase, multiplicative decrease

The combination of the congestion avoidance and fast recovery mechanisms, often called Additive Increase, Multiplicative Decrease (AIMD), leads to a TCP sending rate that is sawtooth shaped. When a small number of TCP flows share a bottleneck link, the senders tend to "synchronize": All the flows increase their sending rate until the traffic exceeds capacity and the bottleneck queue starts to build up. When the queue overflows, packets from all flows are dropped, forcing the flows to halve their window sizes almost simultaneously, and then the process starts over. However, when a large number of flows share a bottleneck, synchronization seems not to happen [3].

Bottleneck queues and RTT variations

The ACK clock partially addresses the problem of overestimation leading to queues building up, since if a queue on the path builds up, the roundtrip time increases, which leads to a decreased sending rate. The ACK clock by itself is however not a very satisfactory control mechanism.

The reason is that in the Internet, there is a quite sharp distinction between bottleneck and non-bottleneck queues. A queue that is not a bottleneck will naturally stay almost empty. On the other hand, the congestion avoidance mechanism of TCP will cause the bottleneck queues to stay close to overflow, at least if there are a large number of flows. Then, the roundtrip time will be a constant value that depends only on *which* queues are bottlenecks, plus a small random noise.

2.2 Wireless links

Wireless links result in disturbances to the IP packet transport that are different from the typical disturbances on wired links. The disturbances on a wireless link may include

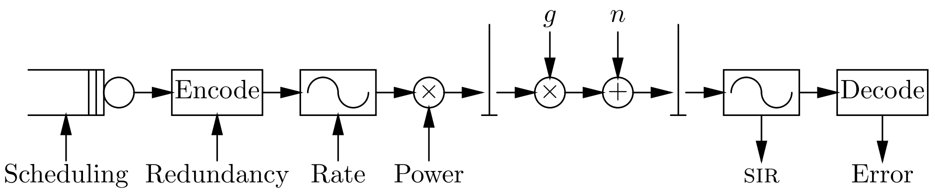

Figure 2.2: Wireless link mechanisms. From left to right: Buffer of frames to send, channel coder, modulator, transmission amplifier, transmission antenna, channel gain, channel noise, reception antenna, demodulator, and channel decoder.

- Packet losses. Unlike wired networks, a wireless link can have a high packet loss rate, and packet losses that are not associated with network congestion.
- Capacity variation. Due to, e.g., movements of the wireless terminal or obstacles, the wireless link may switch between lower and higher sending rates.
- Delay variation. Due to, e.g., local retransmissions, the variation of packet delay may be qualitatively different from that of a wired link.
- Temporary outages. As an extreme form of delay variation, packets may be buffered for a long time, i.e., an RTT or more, during a temporary outage of the radio channel, and be transmitted only after the radio channel is available again.

Automatic control mechanisms of the wireless mechanisms at the link, see Figure 2.2, can be used to reject some disturbances, or trade one type of residual disturbance for another. The signals and the corresponding feedback loops work on different time scales.

Power control

Power control is the fastest loop. SIR-based power control uses feedback from the measured SIR at the receiver to control the transmission power at the sender. This is referred to as the *power control inner loop*, and the control objective is to keep the received SIR close to a reference value SIR_{ref}.

Assume we use a constant sending rate and a constant amount of redundancy for forward error correction (FEC). Let SIR_{bad} denote a SIR at which reception is barely possible, corresponding to, say, a frame loss probability of 50%.

If the inner loop uses a reference value $\text{SIR}_{\text{ref}} > \text{SIR}_{\text{bad}}$, and the fading, i.e., variations of the channel gain, is slow enough, then the inner loop by itself can reject the fading disturbance, and ensure that the received SIR is higher than SIR_{bad} almost all of the time. But when the fading is faster other strategies are needed.

If the fading is too fast for the inner loop, then the variation of the channel gain g is larger than the variation of the power, controlled by the inner loop. The result is that the received SIR will vary in the same way as the channel gain, and at the times when g is smallest, we can have $\text{SIR} < \text{SIR}_{\text{bad}}$.

To deal with this situation, the target value for the SIR can be increased. Then the varying received SIR is essentially increased by a constant. This shortens or eliminates the time intervals when $\text{SIR} < \text{SIR}_{\text{bad}}$. Furthermore, if these time intervals are made short enough, FEC with a suitable amount of redundancy can be used to recover from any demodulation errors.

In Chapter 4 we will consider an outer power control loop that adjusts SIR_{ref} using feedback from the frame error signal from the decoder at the far right in Figure 2.2.

The achievable performance of feedback power control is limited by two main factors: One is the feedback delay, which makes it impossible to track fast changes in the channel gain, as discussed above. The other is limitations on the control signal. Increasing the transmission power causes interference for other users, which coupled to the other users' power control may jeopardize stability of the system: If every user increases his own transmission power without limit, all communication is drowned in interference. Global stability is one of the central issues in distributed power control systems [21], in particular for the uplink in cellular systems.

Forward error correction

Forward error correction adds redundant parity bits to each frame before transmission. On reception, a limited number of errors in the data or parity bits can be corrected. FEC does not rely on feedback, and is therefore not limited by the feedback delay. It can correct errors due to short disturbances of duration down to the transmission time for a single bit. The limit to the effectiveness of FEC is instead in terms of the number of damaged bits per frame.

One technique that is commonly used to increase the effectiveness of FEC over channels with bursts of errors is interleaving: Parity bits are added to each frame, and then transmission of frames are interleaved in time. Then, an error burst corresponding to the transmission time of, say, 100 bits, will not appear as 100 bit errors in a single frame, but as 10 bit errors in each of 10 frames.

FEC is particularly attractive for wireless channels subject to multipath fading. Such channels exhibit sharp dips in the channel gain, which it is not feasible to reject by increasing the transmission power. But since the channel is in such a dip only a fairly small proportion of the time, FEC with suitably chosen parameters can correct the resulting errors. To get the most out of a channel, power control and FEC should be tuned together [6].

Scheduling and link-layer retransmissions

Scheduling is the decision of which frame to send when. For a shared downlink channel, such as the HSDPA channel in the universal mobile telecommunications system (UMTS), the base station is free to decide which user to transmit for in each time slot. The decision can be based on measured or predicted channel quality for the different users.

For an uplink dedicated channel, such as the one considered in Chapter 4, it is not so much a question of when to use the channel, but a question of which frame to send in each time slot. In particular, whether or not a new data frame should be transmitted, or an old frame be retransmitted.

Tuning of the parameters for the link-layer control, i.e., power control, FEC, and rate adaptation, usually tries to optimize the channel throughput, under constraints such as the total cost of base stations and other infrastructure[1]. The selected parameters may correspond to a fairly high frame loss rate, up to 10% [13, 32].

The impact of such a high frame loss rate, on the quality of the services that the link is used for, naturally depends on the type of the service in question. As discussed in the next section, for any service that depends TCP flows, a small loss rate is essential for high quality.

By retransmitting damaged frames, one buys a small loss rate, and pays with increased, random, delays. General recommendations on the use of local retransmissions on Internet links are documented in RFC 3366 [17]. Link-layer retransmissions can be viewed as a feedback loop, driven by the frame error signal. Chapter 4 models one link retransmission scheme, and analyzes the impact on TCP throughput.

2.3 TCP over wireless

Poor TCP performance over wireless links is a well-known problem. The traditional explanation for poor TCP performance is that the wireless link drops packets due to noise and fading on the radio channel, and that TCP interprets all packet losses as indications of network congestion [14]. An overview of the problems and of proposed mechanism at the link or transport layer can be found in [41].

Since TCP performance problems have been observed also on links with link-layer retransmissions, and hence few lost packets, it is not sufficient to understand how TCP is affected by packet losses. Using link-layer retransmissions trades losses for delay, and the resulting delays can also influence TCP negatively. In Chapter 4, we investigate the delay distribution for the uplink in a cellular system, and its interaction with TCP.

Understanding and improving TCP behavior over wireless links is a topic of intense research. This section describes a few representative contributions from each of the main approaches, but it does not attempt to be an exhaustive survey of the field.

[1]This makes the most sense when operators are charging per received bit. With a flat rate business, maybe the tradeoffs would be different?

End-to-end

The end-to-end approach intends to improve TCP behavior over fairly general classes of wireless links.

The Eifel algorithm adds extra information to packets (using the standard TCP timestamp option) to make it possible for the sender to distinguish between acknowledgements for original transmissions and for retransmissions [29]. This information makes it possible for the TCP sender to react more intelligently to certain types of disturbances. In particular, when all packets are delayed for a time longer than the TCP timeout value but not lost (typical for a short temporary outage in a link employing local retransmissions), recovery is improved significantly by using the Eifel algorithm.

The idea of TCP Westwood is to estimate the available bandwidth over the path, based on the timing of received ACKs [30]. When recovering after a loss, this estimate is used to set the new rate to a less pessimistic value than the rate used by standard TCP.

For wireless links where losses are the primary problem, one natural approach is to try to detect if a packet loss is due to congestion or to transmission errors [10, 39, 20]. In general, the end-to-end approach leads to interesting estimation problems, where end nodes use the limited data available, such as ACK timing, to extract information about the network state.

Link-layer

The link-layer approach uses models for the underlying radio, and uses these models for evaluation, tuning, and design, of link-layer mechanisms. In the context of TCP over wireless, common quality measures are user response time and TCP throughput.

Within this class, TCP over links with random errors and no retransmissions is analyzed in [1]. In [27], TCP throughput over a link is simulated, for various radio conditions and link-layer retransmission schemes.

There are many important link tradeoffs that have been investigated, including TCP downlink performance in a WCDMA system with joint rate and power adaptation [22], the tradeoff between link-layer FEC and TCP throughput [12, 5], and the tradeoffs between FEC, ARQ and transmission power [11, 6].

The Gilbert-Elliot model is widely used in the literature; this is a two-state Markov model with one “good” and one “bad” state for the radio channel. TCP behavior over links that use local retransmissions on top of a Gilbert-Elliot radio channel is analyzed in [8, 36],

Cross-layer mechanisms

The end-to-end principle implies that the network and transport layers in a node can not or should not know anything about the link layers in remote parts of the network (the network layer in a node naturally has to be aware about the links that are physically attached to the same node). Architecturally, this is a very sound design, but in some circumstances, in particular when wireless links are involved, it may lead to suboptimal performance.

It is possible to improve the performance by introducing explicit *cross-layer* signalling. This term denotes signalling between layers that are separated both geographically and in networking stack order. Typically, the signalling is between the transport layer in one or both endpoints, and the link layer attached to an intermediate radio link. There are at least two types of cross-layer signalling. The link can inform the transport endpoint of the radio link state, such as the current capacity. One example of such radio network feedback is considered in Chapter 3. It is also possible to let a transport endpoint inform the radio link layer of its requirements, e.g., the preferred tradeoff between loss rate and delay.

Besides the general observation that a controller can usually do a better job the more information it has about the process being controlled, another important reason why cross-layer design is considered, is based on overall deployment issues. Deploying a new version of TCP is a complex and time consuming process. Standardization is fairly slow, there are a large number of TCP-implementations that must be updated, and there are huge number of devices that are attached to the Internet, which must all be updated or replaced until a new version of TCP can replace the current version.

For a solution to the TCP-over-wireless problem to be feasible in the short term, i.e., deployable within a few years, it is essential that it does not require that a majority of Internet devices be upgraded. It is preferable if upgrades are isolated to a certain class of devices, e.g., Internet enabled mobile phones, or to a subset of the network within a single administrative domain, e.g., one operator's network.

Using cross-layer mechanisms provides some additional freedom in the design, which can make a huge difference for the deployability of a solution. Whether or not cross-layer mechanisms are fundamentally needed for a high performance wireless Internet is an open and controversial question [9, 26].

Chapter 3

Radio network feedback

In this chapter, we investigate a cross-layer approach to improve the performance for web browsing (or other forms of TCP *download) over a cellular network.*

3.1 Introduction

When a user downloads a file or a web page from the Internet to a mobile terminal, the path through the network includes a mobile terminal, a radio link to an operator's cellular network, a radio network controller (RNC), the operator's internal core network, a gateway between the operator's network and the Internet, and a web server attached to the Internet. Using a proxy in this way, as illustrated in Figure 3.1, is an example of the split-connection approach to TCP over wireless.

Let us consider the flow of data from the web server to the mobile terminal, using a high bandwidth wireless channel, such as the high-speed downlink packet access (HSDPA) channel in WCDMA. The bandwidth over the radio link varies with time, including outage periods when no communication is possible. Using a pure end-to-end protocol such as TCP, any information about the state of the radio link has to pass down to the terminal and back, adding a significant delay. Furthermore, during a temporary outage, no signalling from the terminal can reach the other endpoint.

This motivates a cross-layer approach: The RNC is well informed about the radio link state, and it is well connected to the Internet. So we can let the RNC transmit radio network feedback (RNF) messages to the sender. A straight-forward application of this idea requires that the TCP stack at the web server is modified to take advantage of the RNF-signalling, which is unrealistic, at least in the short term.

Instead, we use a HTTP proxy in the operator's network.[1] Virtually all web browsers support HTTP proxying, and the proxy communicates with the target web

[1]Using a more transparent TCP proxy is also possible.

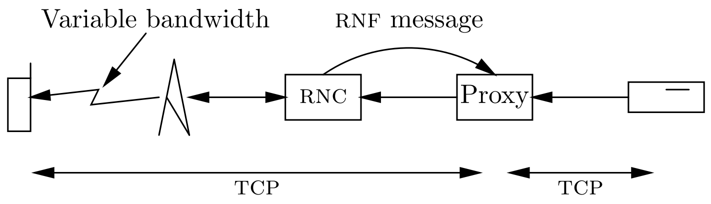

Figure 3.1: Proposed architecture. The mobile terminal on the left downloads a file from the server on the right, via the proxy. During the transfer, the RNC generates RNF messages including information about the current bandwidth over the radio link, and the current RNC queue length. The proxy uses this information to adjust its sending rate.

server using standard HTTP and TCP. Thus, only the RNC and the proxy, both part of the operator's network, need to be aware of the RNF mechanism.

This chapter proposes and analyzes a new control mechanism based on RNF. The performance improvement, compared to end-to-end TCP, is evaluated in `ns-2` simulations. It turns out that the RNF mechanism yields significant improvements for both the radio link utilization and the end-to-end response times experienced by the user. Sections 3.2 and 3.3 describe the system architecture and control structure. The proposed controller is described and analyzed in Section 3.4, and the discrete time implementation is described in Section 3.5. Finally, in Section 3.6, the controller is evaluated using `ns-2` simulations in three realistic use cases.

3.2 Architecture

Consider the transfer of a file, from a server on the Internet, to a mobile terminal attached via an operator's radio access network. Within the operator's network, there are two special nodes: An RNC, which among other things is responsible for allocating bandwidth to user connections, and a proxy, which acts as a gateway between the operator's network and the Internet. The endpoints use standard TCP to communicate with the proxy. The proxy, on the other hand, adapts its sending rate towards the terminal using a custom control algorithm, which is aided by extra information provided by the RNC. We make the reasonable assumptions that the bottleneck for the connection is the radio channel, and that the operator's network, between proxy and RNC, does not suffer congestion.

The architecture is illustrated in Figure 3.1, where two TCP connections have been established: one between the terminal and the proxy and one between the proxy and the server. The RNC sends RNF messages to the proxy, including information about the current bandwidth allocation and the queue length in the RNC. The RNF messages are sent every time the bandwidth changes, and also periodically with a relatively long period time, e.g., one second.

We will compare this system to the nominal setup, in which there is a direct

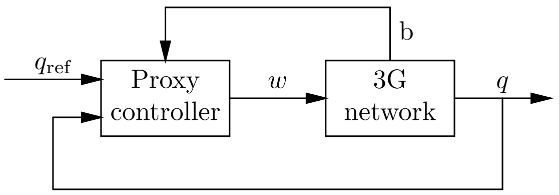

Figure 3.2: Control structure. The controller uses event-triggered feedforward of the available radio link bandwidth b, and time-triggered feedback of the RNC queue length q.

TCP connection between the terminal and the server.

We use the following notation. The bandwidth of the radio link is b. τ_{f} is the time it takes for a packet that is sent by the proxy to reach the RNC, and τ_{b} is the time it takes for a packet that is forwarded by the RNC to reach the terminal, and for the corresponding acknowledgement to get back to the proxy. The corresponding RTT of the cellular network, excluding queueing delay at the RNC, is $\tau = \tau_{\text{f}} + \tau_{\text{b}}$, and the "pipe size", or bandwidth-delay product, is $b\tau$. We assume that b, τ_{f}, and τ_{b} are constant most of the time, with occasional step changes. The queue length at the RNC is denoted q, so that the RTT is $\tau + q/b$. The current window size at the proxy is denoted w.

3.3 Control structure

The objective is to achieve a high utilization of the radio link and maintain the RNC queue close to a reasonably small reference value q_{ref}, by controlling the sending rate of the proxy. Like in standard TCP, we use a window-based algorithm, but unlike standard TCP, we take advantage of explicit information provided by the RNC.

The control signal is the non-negative window size w, which indirectly determines the sending rate. The network provides the proxy controller with two kinds of information. The control structure, with feedforward of radio bandwidth b, and feedback of queue length q, is shown in Figure 3.2.

When the available bandwidth over the radio channel is changed, the RNC sends an RNF message to the proxy to inform it about the new bandwidth. The controller uses an event-triggered feedforward mechanism that takes advantage of this information, and resets the window size to a new value,

$$w = \hat{b}\hat{\tau} + q_{\text{ref}} \tag{3.1}$$

Here $\hat{\tau}$ is an estimate of the propagation delay τ, and $\hat{b}$ is the new bandwidth from the RNC's RNF message.

Between bandwidth updates, the RNC periodically sends RNF messages with the current queue length. The controller compares this feedback information to the reference value and adjusts its window,

$$w_{k+1} = w_k + c(q_{\text{ref}} - q_{k+1}) \tag{3.2}$$

The feedback loop uses a quite long sampling time, on the order of one second, and it is designed to compensate for the bias caused by a feedforward controller based on uncertain information. On shorter time scales, the transmission window is fixed, and transmission is governed by the usual ACK clock.

3.4 Stability analysis

To investigate the stability of the control mechanisms, we use a flow level model. The system input signal is $w(t)$ and the output signal is $q(t)$. The sending rate is related to the window size by

$$r(t) = \frac{w(t)}{\tau + q(t - \tau_{\text{b}})/b}$$

where $\tau + q(t - \tau_{\text{b}})/b$ is the RTT corresponding to the queue length at the time the most recently acknowledged packet was forwarded by the RNC. The rate of change of the queue (as long as it is neither full nor empty) equals the difference between the received rate and the bandwidth,

$$\begin{aligned} \dot{q}(t) &= r(t - \tau_{\text{f}}) - b = \frac{w(t - \tau_{\text{f}})}{\tau + q(t - \tau)/b} - b \\ &= \frac{w(t - \tau_{\text{f}}) - b\tau - q(t - \tau)}{\tau + q(t - \tau)/b} \end{aligned} \tag{3.3}$$

Feedforward control

With feedforward control, the sender uses estimates $\hat{b}$ and $\hat{\tau}$ to compute a constant control signal according to (3.1). Substituting this expression into the queue dynamics (3.3), gives

$$\dot{q}(t) = \frac{\hat{b}\hat{\tau} + q_{\text{ref}} - b\tau - q(t - \tau)}{\tau + q(t - \tau)/b}$$

This time-delayed differential equation has an equilibrium point $q^* = q_{\text{ref}} + \hat{b}\hat{\tau} - b\tau$, and can be rewritten as

$$\dot{q}(t) = -\frac{q(t - \tau_f) - q^*}{\tau + q(t - \tau_f)/b}$$

Define $\tau^* = \tau + q^*/b$ as the RTT corresponding to the equilibrium. The linearized dynamics are

$$\dot{q}(t) = -\frac{1}{\tau^*}(q(t - \tau) - q^*)$$

The Nyquist criterion gives asymptotic stability since

$$\frac{\tau}{\tau^*} < 1 < \frac{\pi}{2}$$

To estimate the convergence time, consider the poles of the linearized system, i.e., the solutions to

$$s + \frac{e^{-\tau s}}{\tau^*} = 0$$

Since the system is stable, all solutions lie in the half plane $\operatorname{Re} s < 0$. Note that for any such pole, it holds that $|s| = e^{-\tau \operatorname{Re} s}/\tau^* > 1/\tau^*$, so the poles lie outside of the disk of radius $1/\tau^*$ around the origin. Let us derive a bound $\lambda > 0$ for the stability exponent of the system, i.e., a bound such that $\operatorname{Re} s < -\lambda$ for all solutions to the characteristic equation above. The trajectories will locally converge to the equilibrium faster than $e^{-\lambda t}$, and $1/\lambda$ is an upper bound for the system time constant. We use the following result.

Lemma 1 *The equation $\alpha x = e^x$, with $\alpha > 0$, has a solution $x > 0$ if and only if $\alpha \geq e$. Furthermore, the solutions satisfy $1/(\alpha - 1) < x < 2\alpha$.*

Proof: See Appendix A. □

First consider a real pool, $s = -x$. Then $-x = -e^{\tau x}/\tau^*$. With $\alpha = \tau^*/\tau$, the first part of the lemma yields $\tau^* \geq e\tau$, and the second part gives $\tau x > 1/(\tau^*/\tau - 1)$, or $x > 1/(\tau^* - \tau)$. So in this case, we have the bound $\lambda > 1/(\tau^* - \tau)$.

Next, consider a complex pole in the left half plane, $s = -x + iy$, with $x, y > 0$. Again we want to find a bound $\lambda > 0$ such that all poles are to the left of $\operatorname{Re} s = -\lambda$. So assume that $x \leq \lambda$. Denote $z = e^{-\tau s} = e^{\tau x}(\cos \tau y - i \sin \tau y)$, and assume that $\tau^*(x - iy) = -\tau^* s = z$, so that s is a pole of the system. First, note that

$$\tau^* y \leq |\tau^* s| = |z| = e^{\tau x} \leq e^{\tau \lambda}$$

Require λ to be such that $e^{\tau\lambda}\tau/\tau^* < \pi/2$, which is always possible for some $\lambda > 0$. Then cos is decreasing on the interval $(\tau y, \pi/2)$. We get for the real part

$$\tau^* \lambda > \tau^* x = \operatorname{Re}(-\tau^* s) = \operatorname{Re} z = e^{\tau x} \cos \tau y > \cos \frac{\tau e^{\tau\lambda}}{\tau^*}$$

This implies that $\lambda > (1/\tau^*) \cos(e^{\tau\lambda}\tau/\tau^*)$. So if we choose λ small enough, this inequality is violated and then there can be no poles with $x \leq \lambda$. Hence, define

$$\lambda' = \{\text{smallest positive solution } \lambda, \text{ to } \tau^*\lambda = \cos \frac{\tau e^{\tau\lambda}}{\tau^*}\}$$

We obtain the general bound for the stability exponent as

$$\lambda = \min\left(\frac{1}{\tau^* - \tau}, \lambda'\right)$$

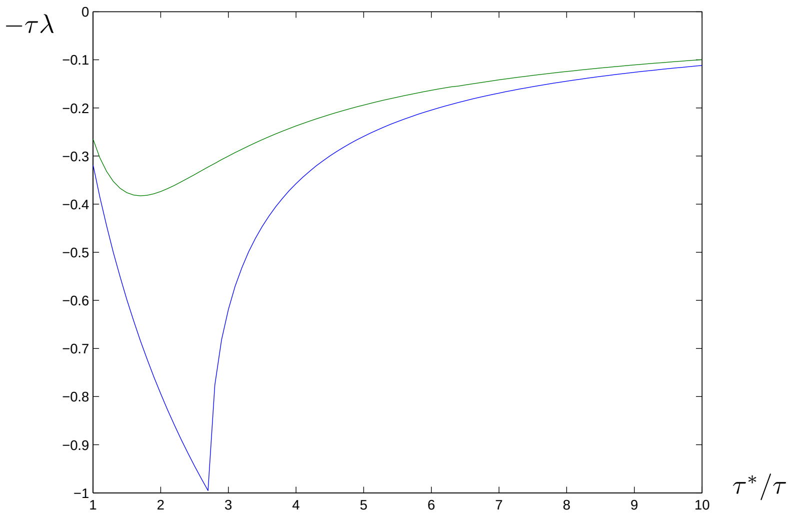

Figure 3.3: Stability exponent $-\lambda$ for $\tau = 1$. The upper curve is the derived conservative estimate $-\lambda = \min(1/(\tau^* - \tau), \lambda')$. The lower curve is computed numerically using an order seven Padé approximation for the delay.

Note that $\tau\lambda$ depends only on the ratio τ^*/τ. Figure 3.3 shows $\tau\lambda$ as a function of τ^*/τ. The convergence speed peeks for $\tau^* = e\tau$, which is when the system has a real double pole at $-1/\tau$. It is easy to obtain the bound $\lambda > 1/(4\tau)$ for the interval $\tau < \tau^* < e\tau$. To summarize, we have the following result.

Proposition 1 *The feedforward control results in an asymptotically stable system, with an equilibrium, which corresponds to a deviation from the reference value q_{ref} that equals the estimation error for the pipe size, $\hat{b}\tau - b\tau$. As long as $\tau^* < e\tau$, the convergence time constant for the linearized system is smaller than 4τ.*

Feedback control

Next, consider feedback control of the window size, using queue length samples. The primary objective of the feedback control is to cancel the bias that results when using feedforward control from estimates $\hat{b}$ and $\hat{\tau}$ that are uncertain. A natural approach is to use an integrating controller,

$$\dot{w}(t) = -c'\big(q(t - \tau_{\mathrm{b}}) - q_{\mathrm{ref}}\big) \tag{3.4}$$

With this control law, the closed loop system is described by the equations

$$\begin{cases} \dot{q}(t) = \dfrac{w(t - \tau_{\mathrm{f}}) - b\tau - q(t - \tau)}{\tau + q(t - \tau)/b} \\ \dot{w}(t) = -c'\big(q(t - \tau_{\mathrm{b}}) - q_{\mathrm{ref}}\big) \end{cases} \tag{3.5}$$

Since q and w are non-negative, these equations are valid only in the interior of the domain $q, w \geq 0$. On the border $w = 0$ we have instead

$$\dot{w}(t) = \max\Big(0, -c'\big(q(t-\tau_{\mathrm{b}}) - q_{\mathrm{ref}}\big)\Big)$$

and similarly for $\dot{q}$ on the line $q = 0$.

The equations have an equilibrium point given by $q^* = q_{\mathrm{ref}}$ and $w^* = b\tau + q_{\mathrm{ref}}$. The RTT in stationarity is $\tau^* = \tau + q_{\mathrm{ref}}/b$. With this notation, the system can be written as

$$\begin{cases} \dot{q}(t) = \dfrac{w(t-\tau_{\mathrm{f}}) - w^* - (q(t-\tau) - q^*)}{\tau^* + (q(t-\tau) - q^*)/b} \\ \dot{w}(t) = -c'(q(t-\tau_{\mathrm{b}}) - q^*) \end{cases}$$

Note that the queue delay influences the system dynamics via the gain in the expression for $\dot{q}$, but the queue is *not* in the signaling path. In the RNF architecture, the RNF messages say how long the queue is, but the RNF messages themselves do not suffer any queueing delay. In other words, the time delays in the differential equations are constant and not depending on the system state.

In the following analysis, we ignore the propagation delay $\tau = \tau_{\mathrm{f}} + \tau_{\mathrm{b}}$. We will show that in this case the system is globally asymptotically stable. Introduce $\widetilde{q} = q - q^*$ and $\widetilde{w} = w - w^*$. Then, the non-linear system without propagation delay can be written as

$$\begin{cases} \dot{\widetilde{q}}(t) = \dfrac{\widetilde{w} - \widetilde{q}}{\tau^* + \widetilde{q}/b} \\ \dot{\widetilde{w}}(t) = -c'\widetilde{q} \end{cases}$$

We make a change of variables by introducing a "virtual time" s, defined by $\mathrm{d}t = (\tau^* + \widetilde{q}/b)\mathrm{d}s$. The virtual time corresponds to the number of RTTs. Then

$$\begin{cases} \dfrac{\mathrm{d}\widetilde{q}}{\mathrm{d}s} = \widetilde{w} - \widetilde{q} \\ \dfrac{\mathrm{d}\widetilde{w}}{\mathrm{d}s} = -c'\tau^*\widetilde{q} - \dfrac{c'}{b}\widetilde{q}^2 \end{cases}$$

These differential equations are still valid only in the interior of the region $w, q \geq 0$, non-negativity has to be enforced at the borders. The phase portrait of this system (for unit parameters) is shown in Figure 3.4. Our goal is to show that this system is globally asymptotically stable in the region $w, q \geq 0$. Define a candidate Lyapunov function

$$V(\widetilde{q}, \widetilde{w}) = \widetilde{q}^2(\widetilde{q} + A) + \frac{Bb}{2c'}\widetilde{w}^2$$

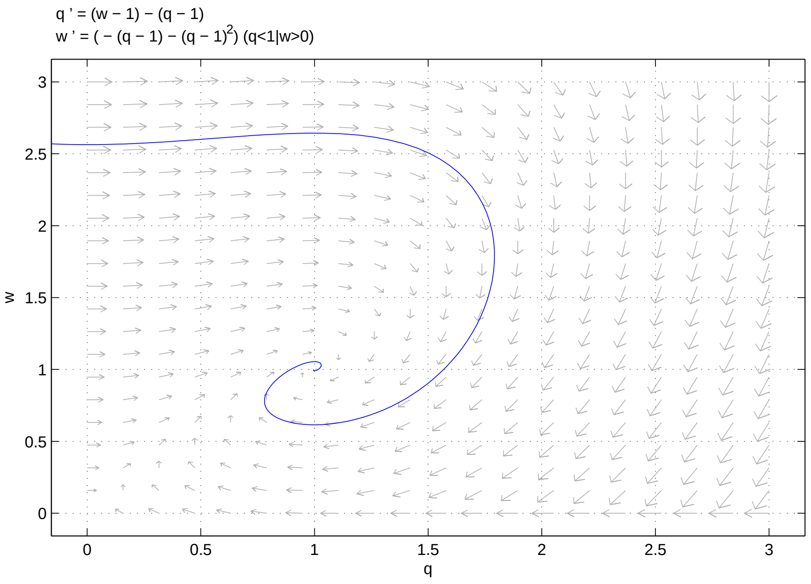

Figure 3.4: Phase portrait of q-w dynamics for the feedback control system.

where A and B are constants to be defined. Then

$$\begin{aligned}\frac{\mathrm{d}V}{\mathrm{d}s} &= \widetilde{q}(3\widetilde{q}+2A)(\widetilde{w}-\widetilde{q})\\ &\quad + B\widetilde{w}\widetilde{q}(-b\tau^*-\widetilde{q})\\ &= -\widetilde{q}^2(3\widetilde{q}+2A)\\ &\quad + \widetilde{q}\widetilde{w}(2A-Bb\tau^*)\\ &\quad + \widetilde{q}^2\widetilde{w}(3-B)\end{aligned}$$

Fix $B=3, A=3b\tau^*/2$. Note that

$$\begin{aligned}\widetilde{q}+A &= q-q_{\mathrm{ref}}+3(b\tau+q_{\mathrm{ref}})/2\\ &= q+q_{\mathrm{ref}}/2+3b\tau/2>0\\ 3\widetilde{q}+2A &= 3(q-q_{\mathrm{ref}})+3(b\tau+q_{\mathrm{ref}})\\ &= 3(q+b\tau)>0\end{aligned}$$

Hence,

$$\begin{aligned}V(\widetilde{q},\widetilde{w}) &= \widetilde{q}^2\left(q+\frac{q_{\mathrm{ref}}+3b\tau}{2}\right)+\frac{3b}{2c'}\widetilde{w}^2\\ \frac{\mathrm{d}V}{\mathrm{d}s} &= -3\widetilde{q}^2(q+b\tau)\le 0\end{aligned}$$

By LaSalle's Theorem, the trajectory converges to an invariant subset of the line $\widetilde{q} = 0$. The only such set is the origin $(\widetilde{q}, \widetilde{w}) = (0, 0)$, corresponding to $(q, w) = (q^*, w^*)$, cf., the point $(1, 1)$ in Figure 3.4. We have thus proved the following result:

Proposition 2 *The feedback control system, with negligible propagation delay in the cellular network, is globally asymptotically stable.*

3.5 Implementation

In practice, the sender does not have access to the continuous-time function $q(t)$, but rely on samples sent from the RNC. Denote the sampling time by T. For the flow model to be valid, an averaging over at least one RTT is needed, so T should not be shorter than the RTT. We next consider the case of a T several times larger than the RTT. (In the simulations presented in next section, the sampling time is one second.)

Between samples, the controller keeps w constant. The queue dynamics will converge to the stationary value within about twice the system time constant. From the analysis for a constant window size, in Section 3.4, we know that the stationary value is $q^* = w - b\tau$. Hence, if T is sufficiently large, and $w(t) = w_k$ for $kT < t \leq (k + 1)T$, then

$$q_{k+1} = q\big((k + 1)T\big) \approx w_k - b\tau$$

The continuous-time integrator controller corresponds to the discrete-time controller

$$w_{k+1} = w_k + c(q_{\text{ref}} - q_{k+1})$$

It turns out that $c = 1$ is a good choice for the controller gain, because then

$$q_{k+2} \approx w_{k+1} - b\tau \approx w_k + q_{\text{ref}} - (w_k - b\tau) - b\tau = q_{\text{ref}}$$

The queue will thus converge to the target value within two samples, $2T$.

3.6 Results

The proposed proxy controller takes advantage of RNF messages and thus gets better performance out of the link compared to standard TCP. The gain has three main components: Connection startup, outage recovery, and bandwidth adaptation. They are discussed in detail below. After that, the RNC queue transients are examined, and at end of the section, we describe a quantitative study for three realistic use cases.

Several advantages with the proxy controller is illustrated by the `ns-2` simulations shown in Figure 3.5, which depicts available bandwidth (dotted), sending rate for proxy controller (dashed), and sending rate for nominal end-to-end TCP (solid). In this figure, note that the transmission via the proxy is completed in about 51 s,

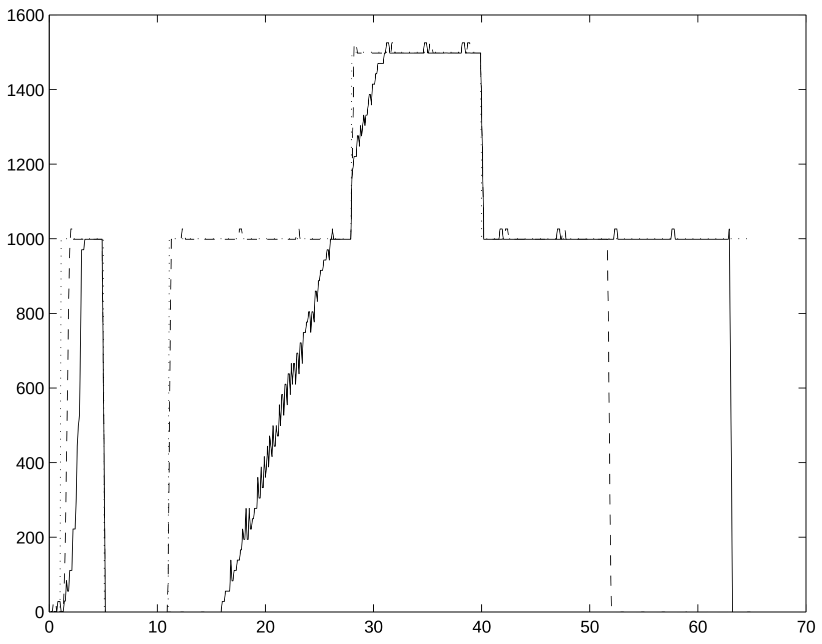

Figure 3.5: Available bandwidth (dotted), sending rate for proposed proxy controller (dashed), and sending rate for nominal end-to-end TCP (solid). The link under-utilization is visible as the area between the bandwidth and the sending rate curves. The sending rates in this and the other figures in this chapter are in kbit/s, averaged over 300 ms. The time axis is in seconds. (That the sending rates sometimes seem to exceed the available bandwidth is an artifact of the sampling of the `ns-2` simulation.)

while the end-to-end TCP transmission lasts approximately 10 s longer. It is clear that the proxy controller is able to utilize the available bandwidth much better than the nominal end-to-end controller.

Connection startup

The connection startup phase is illustrated in the very beginning of Figure 3.5. A zoom in is shown in Figure 3.6. The initial delay before the sending rate starts to rise is the RTT between terminal and server. The rate curve for nominal TCP is governed by the slow start mechanism of TCP (Section 2.1). The flow from the proxy to the terminal does not use slow start at all. However, the flow from the server to the proxy does use slow start, which explains why the rate curve for the proxy can not jump instantaneously to 1 Mbit/s. The reason that the proxy solution outperforms nominal TCP, even though both curves are directly or indirectly limited by the slow

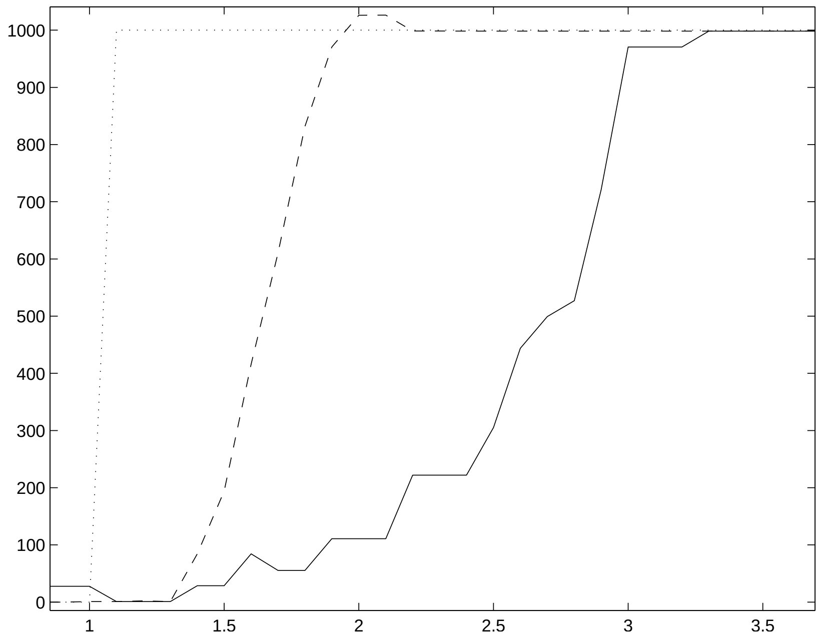

Figure 3.6: Confection startup. The response for the proxy controller is limited by the slow start of the server–proxy connection. It is still much faster than the response of end-to-end TCP.

start mechanism, is that the the RTT between the proxy and the server is much shorter than the RTT between the terminal and the server.

Outage recovery

When an outage occurs, TCP goes into timeout and retransmits lost packets repeatedly using exponential backoff. It will not notice that the outage has ended until it receives an ACK for one of the retransmitted packets, which in the worst case can be 60 seconds after the outage ended. This is illustrated in Figure 3.7, which zoom in the second section (after slow start) of Figure 3.5. Furthermore, when the outage has ended, TCP goes into congestion avoidance mode, where the sending rate is increased only linearly, which means that it takes quite some time until the sending rate approaches the available bandwidth. The proxy solution does not suffer from this additional delay, since it gets explicit information from the RNC when the outage starts ($\hat{b} = 0$) and ends ($\hat{b} > 0$). After the outage ends, the proxy immediately starts sending at the right rate.

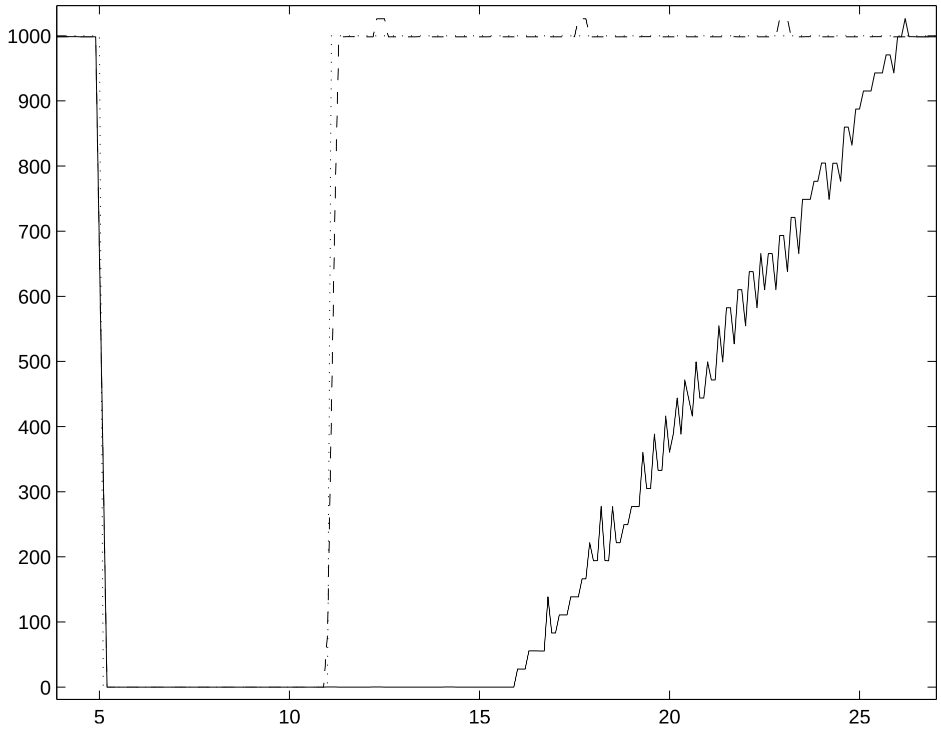

Figure 3.7: Outage recovery. Since the proxy controller does not use exponential backoff or congestion avoidance, it recovers much faster than the nominal TCP from an outage.

Bandwidth adaptation

When the bandwidth of the radio channel is increased, a TCP sender in congestion avoidance mode increases its sending rate slowly. When the bandwidth is decreased, the TCP sender will not notice until some packet is lost due to buffer overflow in the RNC. (In our simulations this does not happen, since we have a very large RNC buffer.) Thanks to the feedforward mechanism in the proxy controller, it reacts more quickly, as illustrated in Figure 3.8, which is a zoom in of the middle section of Figure 3.5. The performance gain is not as significant as the gains in the slow start and outage recovery phases.

RNC queue transient

The size of the RNC queue is time-varying. When the bandwidth of the radio channel is changed, the queue will initially drift off from the reference value, cf., the analysis in Section 3.4. The magnitude of the variation depends on the propagation delay

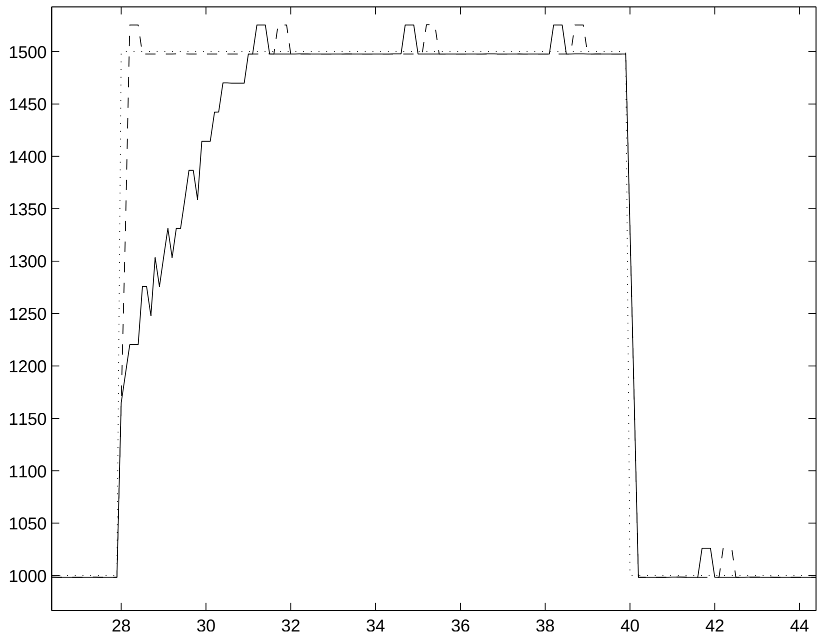

Figure 3.8: Bandwidth adaptation. The proxy controller reacts faster to radio bandwidth changes than the nominal TCP controller.

for the RNF messages. A typical value of the propagation delay is 50 ms, but can depend on the implementation as well as dynamic behavior. Figure 3.9 shows the evolution of the queue length when the bandwidth is changed. It indicates what is a reasonable size of the RNC buffer.

Quantitative evaluation

In order to evaluate the performance of the proposed proxy controller in some practical situations, a set of use cases was defined and a `ns-2` simulation study was performed for each of them. The proxy setup in Figure 3.1 is compared to a nominal setup, in which there is no proxy but only a direct TCP connection between the terminal and the server. The links between the server and the RNC are set to 2 Mbit/s. The wireless link between the RNC and the terminal has a lower time-varying bandwidth and a one-way delay of 60 ms. We also use an average RTT (server-terminal-server) of 300 ms. The slow start threshold was set to 50, to ensure that the TCP sender does not enter the congestion avoidance phase prematurely. Figure 3.5 illustrates the transmission of a 5.7 Mbyte file.

The performance of the proxy setup was evaluated in terms of the average link utilization and the time to serve a user (TTSU). The TTSU is defined as the time

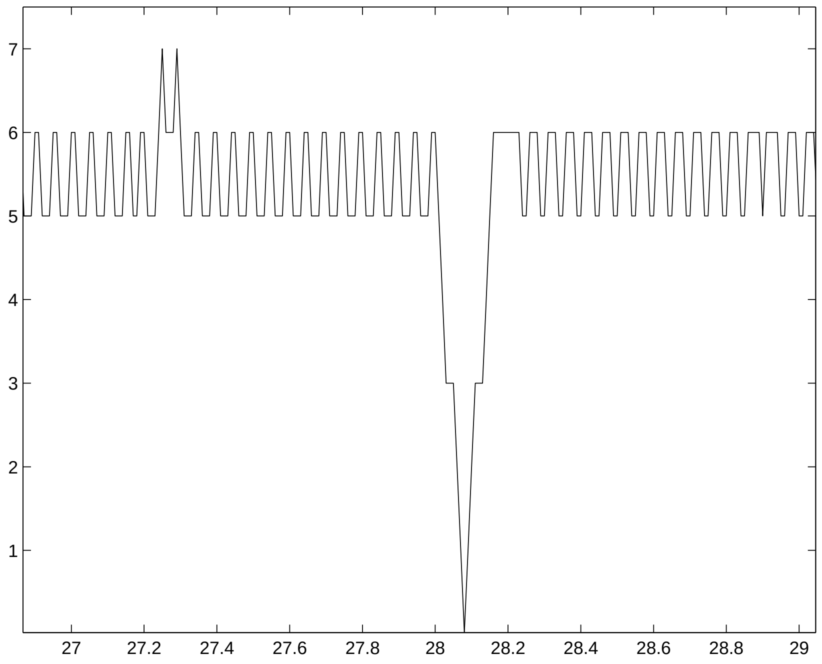

Figure 3.9: Evolution of the queue length, when the bandwidth is increased at 28 s in the simulation. The target value for the queue length is 6 packets.

elapsed from when the connection is established (SYN packet sent by the requester) until the last data packet is received at the terminal. We consider three use cases, aimed at modeling common services to be provided over a WCDMA system. Use case 1 models a 4 Mbyte file transfer. Use case 2 imitates web browsing and is based on statistical data [35, 23]. In this use case, three files (whose sizes vary between 51.5 Kbyte and 368.5 Kbyte) are downloaded to a terminal during one web session. Use case 3 models a large (25 Mbyte) file transfer. The results are summarized in Table 3.1. The TTSU is particularly improved with the proxy controller for relatively small file downloads (such as web browsing). Link utilization is close to 100% for the proxy controller for large files.

		TTSU [s]	Utilization [%]
Use case 1	Nominal	47.3	77%
	Proxy	39.2	98%
Use case 2	Nominal	3.9	51%
	Proxy	3.0	78%
Use case 3	Nominal	233.6	61%
	Proxy	220.5	98%

Table 3.1: Comparison of time to serve a user (TTSU) and link utilization for three use cases.

3.7 Summary

TCP-based applications in a WCDMA system were studied in two setups: a nominal one that employs end-to-end TCP Reno and a new one with an intermediate proxy. The proxy solution uses RNF messages from the data-link layer in the RNC to the transport layer in the proxy. The proxy uses feedforward control to adjust to a varying bandwidth, and feedback control to maintain the RNC queue length close to a desired value. It was shown that the resulting closed-loop system is stable and that the convergence time is linked only to the propagation delay in the cellular system. The performance gain was estimated using `ns-2` simulations. In these evaluations, we considered three realistic use cases. The proxy controller was able to reduce the time to serve users and increase the radio link utilization. The proposed control architecture might substantially improve the user experience of wireless Internet. The proposal is transparent to both endpoints, so no modifications need to be done to the terminal or the server.

Chapter 4

TCP over a power controlled channel

We develop a model for a cellular link, including processes from the control of the transmission power up to the end-to-end congestion control.

4.1 Introduction

When studying TCP over wireless, the model for the channel loss process is of central importance. Since it is difficult to accurately describe all factors influencing the radio transmission, analysis of the wireless system has to be based on models that are simplified. When using a model for control design, even crude models often work remarkably well, but it is of course necessary that the model captures the range of dynamics we wish to control.

In the literature, it is common to focus on the frame loss process, which is often described using the Gilbert-Elliot model. The Gilbert-Elliot model is a two-state Markov model, corresponding to one good and one bad state for the radio channel. Each state is associated with a constant loss probability.

In this chapter, we depart from the Gilbert-Elliot model, by looking at a realistic model of a power controlled channel [40]. We assume that the channel uses an inner loop power control that is able to keep the signal to interference ratio (SIR) close to a reference value, viewing the residual control error as noise. The loss process of the power controlled channel is generated by the outer loop power control, which adjusts the reference value based on experienced losses.

When deploying UMTS, which uses power controlled channels, the target value for the resulting frame loss rate can be on the order of several percents. The wireless link employs local retransmissions of damaged frames. These link-layer retransmissions transform a link with constant delay and random losses into a link with random delay and almost no losses.

TCP was designed for wired networks, where packet losses are usually due to

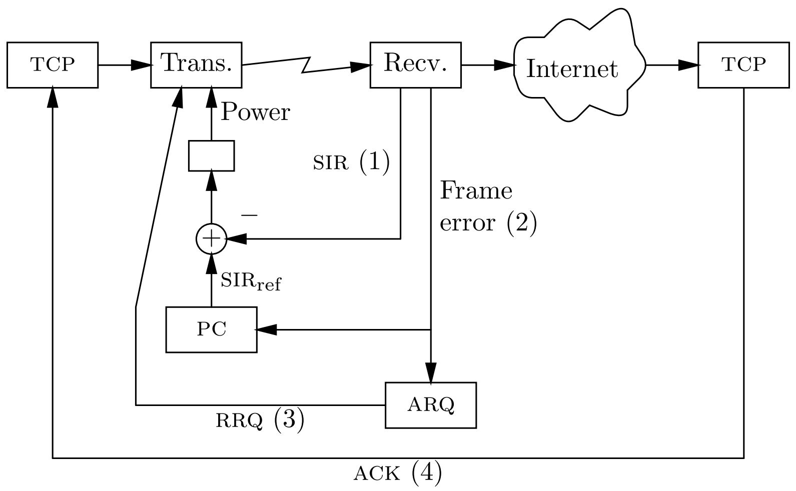

Figure 4.1: System overview: A terminal (on the left), attached to a wireless link, transmits data to a receiving node attached to the wired Internet. There are four feedback loops affecting the delay distribution and throughput: Inner loop power control (1), outer loop power control (2), link-layer retransmissions (3), and end-to-end congestion control (4).

network congestion, not transmission errors, and TCP does not work well with high packet loss rates. But achieving a small loss rate comparable to that of a wired link is not sufficient for TCP to work as well over wireless links as over the wired Internet; performance problems have been observed also with wireless links that use local retransmissions. As seen from the TCP end points, the difference between such a link and a wired one is no longer the loss rate, but the packet delay distribution.

In this chapter we derive the packet delay distribution of a power controlled channel, the probabilities of triggering spurious timeout and spurious fast retransmit in TCP, and the corresponding degradation in TCP throughput. For concreteness, particular choices of various model parameters are made throughout the analysis, but it is straight-forward to apply the same analysis for other choices of parameters.

The results in this chapter seem to confirm recent simulation studies using more detailed models for the wireless link [27]. Much of our analysis focus is on the distribution of the end-to-end roundtrip delay, so we would like to compare our results to empirical studies of this distribution, for both wired and wireless networks. For the wired case, one study found that the RTT distribution can be modelled as a shifted Gamma-distribution [34]. Empirical studies of the RTT distribution for wireless links seem to be rare, and also for the wired Internet, the delay distribution is not well understood.

As a motivation for our work, we believe that the engineering freedom we have

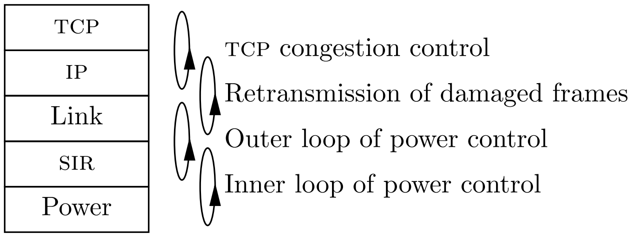
TCP
IP
Link
SIR
Power
TCP congestion control
Retransmission of damaged frames
Outer loop of power control
Inner loop of power control

Figure 4.2: Several layers in the networking stack interact. Feedback control is used to reduce the influence lower layers have on layers above, and thereby support layer separation.

in designing the link-layer processes, such as the scheduling of retransmitted frames, should be used to make the link friendlier to TCP. The next chapter discusses initial work along this line. To be able to design the link-layer mechanisms in a systematic way, good models for the link-layer processes, and their interaction with TCP, are essential.

When using TCP over a wireless link, there are several interacting control systems stacked on top of each other, as illustrated in Figures 4.1 and 4.2. At the lowest layer, the transmission power is controlled in order to keep the signal to interference ratio (SIR) at a desired level. This is a fast inner loop intended to reject disturbances in the form of "fading", or varying radio conditions. On top of this, the outer power control loop tries to keep the frame error rate constant, by adjusting the target SIR of the inner loop. Next, there are local, link-layer, retransmissions of damaged frames. Finally, we have the end-to-end congestion control of TCP.

4.2 Radio model

Data is transmitted over a radio link, e.g., the WCDMA radio link used in UMTS, as a sequence of *radio frames*. One radio frame corresponds to a *transmission time interval* (TTI) of 10 or 20 ms. Depending on the bandwidth (typically from 64 kbit/s to 384 kbit/s) the size of a frame can vary from 160 octets (the small 10 ms TTI is not used for the lowest data rates) to 960 octets.

The transmission of the radio frames is lossy. Let p denote the overall probability that a radio frame is damaged. The power of the radio transmitter is controlled, so that the loss probability stays fairly constant. The target frame error rate, p_{ref}, is a deployment tradeoff between channel quality and the number of required base stations. For UMTS the target frame error rate is often chosen to be about 10% [13, 32]. In the following, we thus assume $p_{\text{ref}} = 0.1$.

Power control

The typical SIR-based power control uses an inner loop that tries to keep the SIR close to a reference value SIR_{ref}. This loop, marked (1) in Figure 4.1, often has a sample frequency of 1500 Hz, and a one bit feedback that is subject to a delay of two samples, i.e., 1.3 ms. The inner loop is often able to track the reference SIR_{ref} within 2-3 dB, with a residual oscillation due to the delay and the severe quantization of the power control commands in the inner loop [21]. The period of this oscillation is typically less than 5 samples, i.e., 3.3 ms.

As there is no simple relationship between the SIR and the quality of the radio connection, there is also an outer loop that adjusts SIR_{ref}. The SIR-updates are controlled by the block "PC" in Figure 4.1. This loop uses feedback from the decoding process; in this thesis we assume that the power control outer loop is based on frame errors. The outer loop uses a sampling interval of one TTI, one order of magnitude slower than the inner loop. In control theory, such use of a fast inner loop and a slower outer loop is called a cascaded control system.

As it is hard to estimate the frame error rate accurately, in particular if the desired error rate is small, one common approach is to increase SIR_{ref} significantly when an error is detected, and decrease SIR_{ref} slightly for each frame that is received successfully. It is interesting to note that this strategy resembles the TCP "additive increase, multiplicative decrease" congestion control strategy. In the next section, we discuss this approach in more detail.

Markov model

The outer loop of the power control sets the reference value for the SIR. Given a particular reference value $\text{SIR}_{\text{ref}} = r$, the obtained SIR can be modelled as a stochastic process. Together with the coding scheme for the channel, we get an expected probability for frame errors. If the coding scheme is fixed, the probability of frame errors is given by a function $f(r)$ which can be computed from models of the channel and coding. Let us consider binary phase shift keying (BPSK). Then we get the approximation

$$f(r) \approx 1 - \left(1 - Q(\sqrt{2\alpha e^{\beta r + \beta^2\sigma^2/2}})\right)^L$$

where α is the coding gain, $\beta = \ln 10/10$, σ is the standard deviation of the received SIR, L is the number of bits in a radio frame, and $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-x^2} \mathrm{d}x$ [40]. We take $\alpha = 4$, also from [40], $\sigma = 1$ dB, corresponding to a small SIR oscillation as in [21], and $N = 2560$, corresponding to a modest link capacity of 128 kbit/s. This function $f(r)$ is shown in Figure 4.3.

In general, $f(r)$ is a decreasing, threshold-shaped function. For small enough r, $f(r) \approx 1$, i.e., almost all frames are lost. For large enough r, $f(r) \approx 0$, i.e., almost no frames are lost. The operating point of the power control is the point close to the end of the threshold, where $f(r) = p_{\text{ref}}$, marked with an asterisk in Figure 4.3.

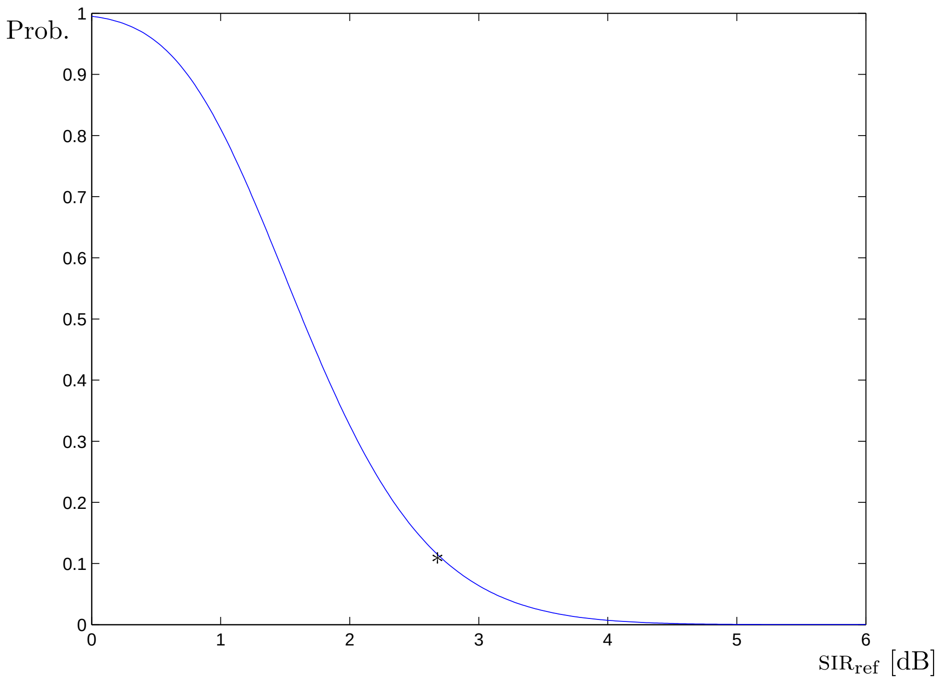

Figure 4.3: Frame loss probability as a function of SIR_{ref}. The star marks the operating point $f(r) = p_{\text{ref}}$.

It is the shape of $f(r)$ close to this point that is of primary importance for the power control behavior. Note that for control purposes only a crude approximation of the true $f(r)$ is necessary.

The outer loop of the power control uses discrete values for SIR_{ref}. One way to keep the frame error probability close to the desired probability p_{ref} is to change SIR_{ref} by fixed steps, based on a fixed step size Δ. Whenever a radio frame is received successfully, SIR_{ref} for the next frame is decreased by Δ. Whenever a radio frame is damaged, SIR_{ref} for the next frame is increased by $K\Delta$. This mechanism yields an average of one lost frame out of $1 + K$, which implies $p = 1/(1 + K)$. We consider the case that K is a positive integer, and in particular, $K = 9$ which implies $p = 0.1 = p_{\text{ref}}$. The value of Δ is an important control parameter, which influences the performance of the power control. For simplicity, we consider a fixed Δ, even if some systems may allow the step size Δ to be adjusted in real time.

In [40], the varying SIR_{ref} was modelled as a discrete Markov chain. The mean and variance of the received SIR was derived, and the dependency on the power control step size Δ and on the inner loop performance was investigated. We use the same Markov model, but focus on the resulting frame error process and its implications for TCP.

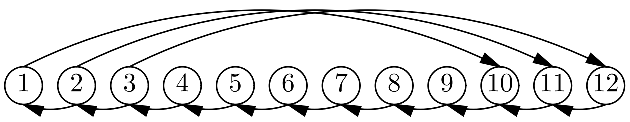

Figure 4.4: Power control Markov chain, for $N = 12$ and $K = 9$. Each state k corresponds to a SIR_{ref} value of $r_0 + k\Delta$. There are at most two possible transitions from state k: to $k + 9$ and to $k - 1$.

To obtain a *finite* Markov chain for a given function $f(r)$ and a step size Δ, we first modify f by setting

$$f(r) = \begin{cases} 1 & r < r_{\min} \\ 0 & r > r_{\max} \end{cases}$$

Figure 4.4 illustrates the case $N = 12$ and $K = 9$. State $k, 1 \le k \le N$, corresponds to $r = r_0 + k\Delta$, with r_0 and N chosen such that $r_0 + \Delta < r_{\min}$ and $r_0 + (N - K + 1)\Delta > r_{\max}$. Define $f_k = f(r_0 + k\Delta)$. Then $f_k = 1$ for $k \le 1$ and $f_k = 0$ for $k > N - K$ (for convenience, we define f_k for all integers k, even if it is $1 \le k \le N$ that is of primary interest). The transitions from state k are as follows: If $k = 1$, then the next state is always $1 + K$. If $2 \le k \le N - K$, then the next state is $k + K$ with probability f_k (transmission fails), and otherwise $k - 1$ with probability $1 - f_k$ (transmission succeeds). Finally, if $N - K < k \le N$, then the next state is always $k - 1$.

Let π_k denote the stationary probability for state k. It is straightforward to compute $\pi_k, k = 1, \ldots, N$, as the solution to the linear equations

$$\begin{aligned} \pi_k &= (1 - f_{k+1})\pi_{k+1} && 1 \le k \le K \\ \pi_k &= f_{k-K}\pi_{k-K} + (1 - f_{k+1})\pi_{k+1} && K < k \le N - 1 \\ \pi_N &= \pi_{N-K} f_{N-K} \\ \sum_{k=1}^{N} \pi_k &= 1 \end{aligned}$$

Figure 4.5 shows the stationary distribution for three values of Δ. Here, $r_0 = r_{\min} = 0$ dB, $r_{\max} = 6$ dB, $K = 9$, and $N = 1 + 6/\Delta$. Note that a smaller step size gives a more narrow distribution, and a smaller average SIR, which means that there is a tradeoff between energy efficiency and short response times. When the radio conditions for one user change so that the user needs a higher SIR, then a small Δ implies that the power control will adjust slowly, and in the mean time, the user will experience a poor quality. On the other hand, a large Δ implies a higher average transmission power for all users, and hence more interference between users. This interference reduces the number of users that can be served simultaneously, i.e., the capacity of the cell.

The threshold function $f(r)$ is also shown in Figure 4.5 (rescaled to fit). For a large Δ, the power control state, i.e., SIR_{ref}, will move quite far along the tail

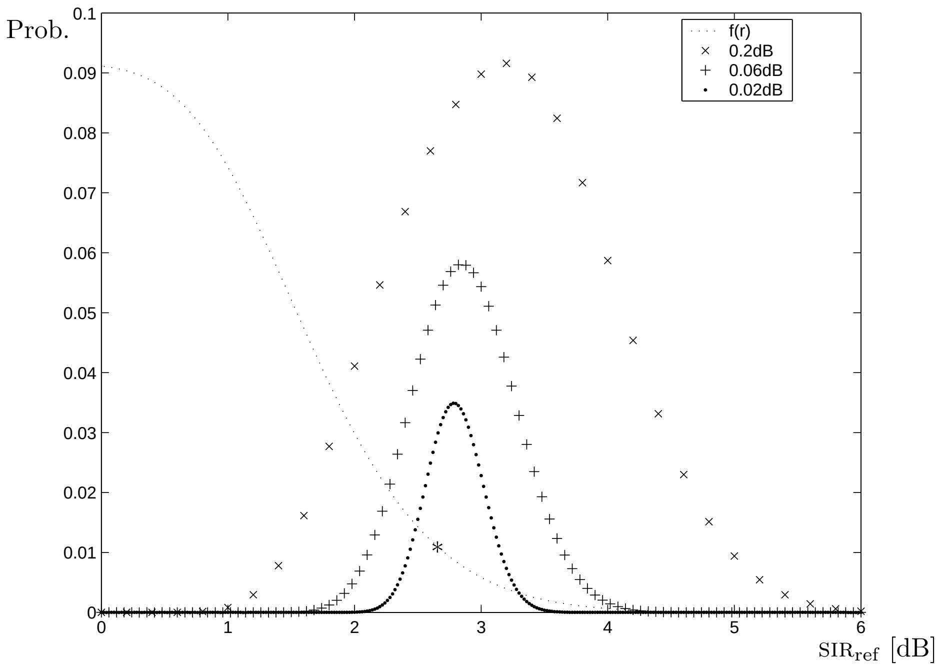

Figure 4.5: Stationary distribution for the power control. Each mark represents one state of the power control, the corresponding value of SIR$_{\mathrm{ref}}$, and its stationary probability π_i. The dotted curve is the threshold-shaped function $f(r)$, scaled to fit in the figure, which represents the frame error probability as a function of SIR$_{\mathrm{ref}}$. The star marks the desired operating point.

of the $f(r)$ threshold. For a smaller Δ, the control state will stay close to the operating point where $f(r) = p = 0.1$. In the latter case, a linearization of $f(r)$ about the operating point gives a good approximation of the dynamics generated by the Markov chain.

Radio frame loss process

Let e_t denote a realization of the radio frame loss process: $e_t = 0$ if the frame sent in time slot t is received correctly, and $e_t = 1$ if the frame is damaged. Define

$$s_t = 1 - t + (K+1)\sum_{i=1}^{t-1} e_i$$

Then s_t is the change in power control state from time 1 to time t, and it is thus just another way to express that the state moves K steps upward when a frame is lost, and one step downwards when a frame is received successfully.

Consider a finite segment of the loss process, $e_1, e_2, \ldots, e_\ell$, of length ℓ. If we assume that at $t = 1$, the initial state of the power control is distributed according to the stationary distribution π_i, then the probability that the realization $e_1, \ldots e_\ell$ occurs can be computed by conditioning on the initial state i:

$$\mathrm{P}(e_1, \ldots, e_\ell) = \sum_{i=1}^{N} \pi_i \prod_{t=1}^{\ell} \left(e_t f_{i+s_t} + (1 - e_t)(1 - f_{i+s_t}) \right)$$

We have done an exhaustive calculation of these probabilities for all loss sequences of length up to 20. In the following sections, we will use these probabilities to investigate the experience of an IP packet, or a sequence of IP packets, that traverses the link.

4.3 Link-Layer Retransmissions

The simplest way to transmit IP packets over the wireless link is to split each IP packet into the appropriate number of radio frames, and drop any IP packet where any of the corresponding radio frames were damaged. But as is well-known, TCP interprets all packet drops as network congestion, and its performance is therefore very sensitive to non-congestion packet drops. An IP packet loss probability on the order of 10% would be highly detrimental.

There are several approaches to recover reasonable TCP performance over wireless links. In this chapter we concentrate on a local and common mechanism: Automatic repeat request (ARQ). The link receiver detects frame damage, and requests that the damaged frame be retransmitted over the link. Use of ARQ is an option in standard wireless network protocols, see [4] for an evaluation of these options in the IS-2000 and IS-707 RLP standards. The effect of link-layer retransmissions is to transform a link with constant delay and random losses into a link with random delay and almost no losses.

There are several possible ARQ schemes. We will consider one of the simpler, the $(1, 1, 1, 1, 1)$-negative acknowledgement scheme [27], which means that there are five rounds, and in each round there is a single retransmission request. When the receiver detects that the radio frame in time slot t is damaged, it sends a retransmission request to the sender. The frame is scheduled for retransmission in slot $t + 3$ (where the delay 3 is called the RLP NAK guard time). If also the retransmission results in a damaged frame, a new retransmission request is sent and the frame is scheduled for retransmission in slot $t + 6$. This goes on for a maximum of five retransmissions.

In the next section, we put together the frame loss process and the retransmission scheduling, to derive IP layer properties of the link. The same methods can be applied to other retransmissions schemes than the $(1, 1, 1, 1, 1)$-scheme considered here.

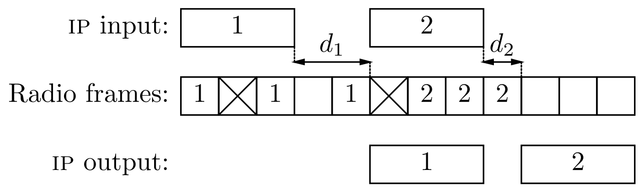

Figure 4.6: Overlaying IP transmission on top of the frame loss process and frame retransmission scheduling. Two IP packets, corresponding to $n = 3$ radio frames each, are transmitted over the radio link. The second and sixth frames are damaged, crossed out in the figure, and scheduled for retransmission three frames later. The resulting retransmission delay for the i:th IP packet is denoted d_i.

4.4 TCP/IP Properties

When transmitting variable size IP packets over the link, each packet is first divided into fix size radio frames. We let n denote the number of radio frames needed for the packet size of interest. For the links we consider, we have $n \leq 10$, where $n = 10$ is used if we send packets of 1500 octets over a slow link with the smallest frame size of 160 octets.

The outermost feedback loop we consider is the end-to-end congestion control of TCP, marked (4) in Figure 4.1. The role of TCP is to adapt the sending rate to the available bandwidth. The inputs to the TCP algorithm are measured roundtrip time and experienced loss events, both of which depend on the links traversed by the TCP packets. To understand and predict the behavior of TCP, we must consider how it interacts with the power control and the link-layer retransmissions.

For any sequence of incoming IP packets, and for any finite loss realization, we can overlay the IP packets on top of the frame sequence to see which frames are retransmitted, and when each IP packet is completed. This process is illustrated in Figure 4.6. Our results for properties of interest, such as the IP packet delay distribution, and the probability of reordering events, are thus obtained by conditioning on the realization of the frame loss process, and a summation over all possible loss sequences of some finite length. Sequences of length 20 have been sufficient for our purposes, in the sense that the input sequences we have considered are short enough that they, with high probability, can be successfully transmitted within 20 frames.

IP packet delay

One important link property for TCP is the delay distribution of packets traversing the link. Using the loss sequence probabilities from Section 4.2, we can explicitly compute the IP packet delay distribution. The distribution for $\Delta = 0.06$ dB and increasing values of n is illustrated in Figure 4.7. The expected value and standard deviation for three values of the step size are shown in Table 4.1. Only the delay

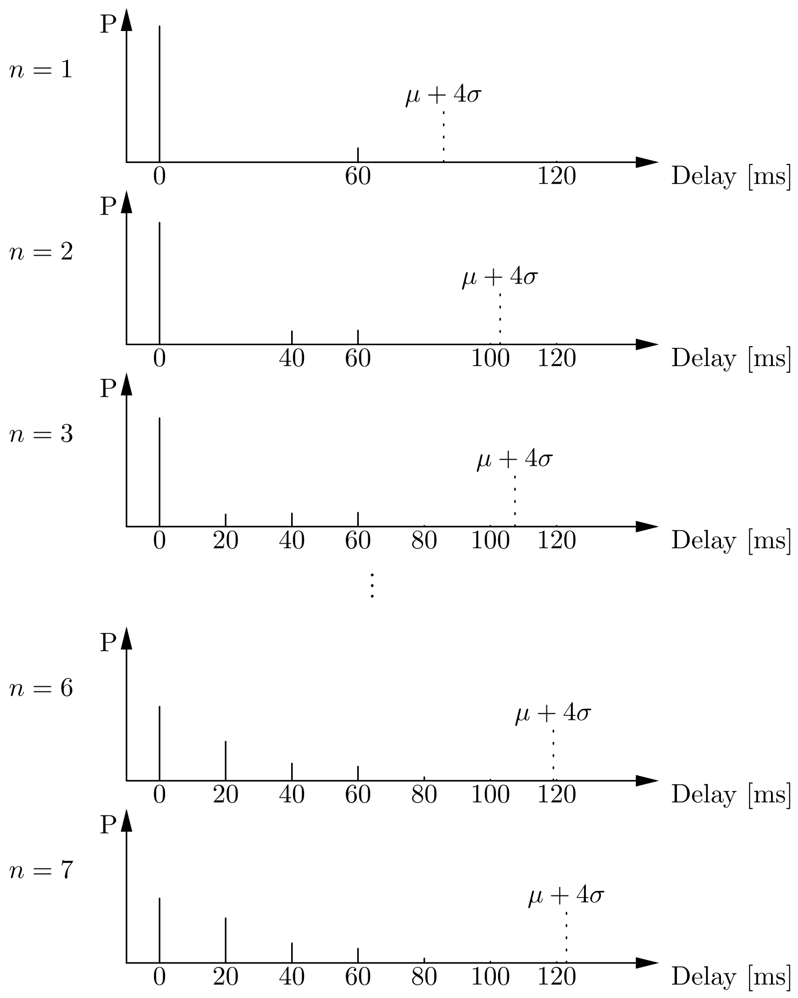

Figure 4.7: Retransmission delay distribution. Each bar shows one possible retransmission delay, and the corresponding probability. These probabilities are calculated for $\Delta = 0.06$ dB, and for packet sizes $n = 1, 2, 3, 6, 7$. The TTI is 20 ms. For each distribution, the value of $\mu + 4\sigma$ is also shown; the tail to the right of this mark is related to spurious TCP timeout events.

	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
n = 1	0.31±0.94	0.32±0.99	0.33±1.03
2	0.51±1.08	0.53±1.15	0.54±1.20
3	0.62±1.08	0.64±1.18	0.65±1.24
4	0.72±1.09	0.74±1.21	0.76±1.28
5	0.83±1.09	0.85±1.23	0.87±1.31
6	0.94±1.09	0.96±1.25	0.98±1.35
7	1.06±1.09	1.07±1.27	1.09±1.38
8	1.17±1.09	1.18±1.29	1.19±1.41
9	1.28±1.09	1.29±1.30	1.30±1.44
10	1.39±1.09	1.40±1.31	1.41±1.46

Table 4.1: Mean and standard deviation of the IP packet delay distribution, measured in units of TTI.

due to retransmissions is included (an n-frame packet is of course also subject to a propagation delay of n TTI).

The mean delay increases almost linearly with the packet size n, which is natural since the expected number of damaged frames is proportional to n. The standard deviation increases more slowly; one explanation is that the "repairing power" of a single retransmitted frame is fairly large, independent of the packet size.

The step size Δ seems to mostly influence the deviation, not the mean. There is also a significant dependence on Δ when looking at individual probabilities, i.e., the individual bars in Figure 4.7.

TCP fast retransmit

If a packet in a TCP stream is delayed so much that it lags behind three other packets (see Figure 4.8), the acknowledgements for the next three packets are "duplicated", and the sender will go into fast retransmit/fast recovery mode. That means that from the point of view of TCP and its congestion control, the packet is lost.

Computing the precise number of fast retransmit events is not trivial. However, we can estimate the probability that fast retransmissions occur as follows. Consider a randomly chosen time, when the power control state is governed by the stationary distribution π_i. Assume that four IP packets, each of size n radio frames, arrive to the radio link back-to-back. We then consider all possible loss sequences, and sum the probabilities of the sequences which cause the first packet to be delayed long enough to be the *last* of the four packets to be received successfully and completely. This gives us the probability of spurious fast retransmit, denoted P_{FR}. Our results are displayed in Table 4.2.

The probability decreases rapidly as the packet size increases. To understand why, consider the delay d, measured in number of frames, suffered by the first packet in Figure 4.8. A necessary condition for the packet to lag behind the next three

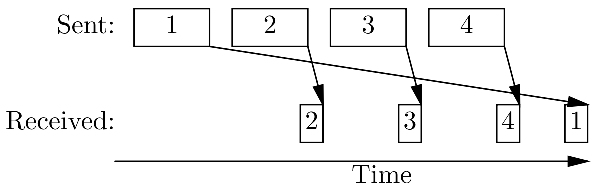

Figure 4.8: Reorder triggers fast retransmit.

packets, is that $d \geq 3n$. For all n, the probability $\mathrm{P}(d > x)$ decays rapidly with x, while both the mean and standard deviation for the delay grows only slowly with n (see Table 4.1). Then $\mathrm{P}_{\mathrm{FR}} \leq \mathrm{P}(d > 3n)$ must also decay rapidly with n.

Therefore, spurious fast retransmissions will not be a problem for larger packet sizes. We also see a clear difference between small and large step sizes Δ, this influence is discussed further in Section 4.5.

Timeout

A timeout event occurs when a packet, or its acknowledgement, is delayed too long. Let $\textsc{rtt}_k$ denote the roundtrip time experienced by packet k and its corresponding acknowledgement. The TCP algorithm estimates the mean and deviation of the roundtrip time [24]. Let $\widehat{\textsc{rtt}}_k$ and $\widehat{\sigma}_k$ denote the estimated roundtrip time and deviation, based on measurements up to $\textsc{rtt}_k$. TCP then computes the timeout threshold for the next packet as

$$\textsc{rto}_{k+1} = \widehat{\textsc{rtt}}_k + 4\widehat{\sigma}_k$$

which means that the probability that packet k causes a timeout is given by

$$\mathrm{P}_{\mathrm{TO}} = \mathrm{P}(\textsc{rtt}_k > \textsc{rto}_k)$$

We assume that the values $\textsc{rtt}_k$ are identically and independently distributed according to the delay distribution given in Section 4.4. For simplicity, we also assume that the estimates $\widehat{\textsc{rtt}}_k$ and $\widehat{\sigma}_k$ are perfect and equal to the true mean and standard deviation of $\textsc{rtt}_k$. Under these assumptions, $\textsc{rto}_k$ is constant, and we can omit the subscript.

In TCP, timeout is the last resort recovery mechanism, and in order to get reasonable performance, a timeout must be a very rare event. The value of P_{TO} will of course depend on the delay distribution. For a Gaussian or uniform distribution, P_{TO} is very small (this is discussed further in the next chapter). Calculating the P_{TO} from the delay distribution of Section 4.4 results in the probabilities in Figure 4.9. We see that the probability for spurious timeout is significant for all packet sizes.

The dependence on step size and packet size looks complicated, but can be understood from Figure 4.7. The P_{TO} is the tail of the distribution to the right

$\mathrm{P_{FR}}$	$\Delta = 0.2$	$\Delta = 0.06$	$\Delta = 0.02$
n = 1	0.24%	0.63%	0.79%
2	0.00%	0.02%	0.05%
3	0.00%	0.00%	0.00%
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 4.2: Probability of spurious fast retransmit.

of the $\mu + 4\sigma$ mark in Figure 4.7. When n is increased, each of these probabilities grows slowly, but at the same time the $\mu + 4\sigma$ mark moves to the right. For $n \leq 6$, the $\mathrm{P_{TO}}$ is the sum of the probabilities for 120 ms delay and up, and each of these grow with n. But for $n = 7$, $\mu + 4\sigma > 120$, so the bar at 120 ms no longer contributes to the $\mathrm{P_{TO}}$. And since the probability for a 120 ms delay is approximately 0.6%, the $\mathrm{P_{TO}}$ drops by 0.6 percentage points when n is increased from 6 to 7, as can be seen in Figure 4.9 for $\Delta = 0.06$ dB.

4.5 Throughput Degradation

In this section, we derive an estimate for the performance degradation due to either of spurious fast retransmissions and spurious timeouts. The relative degradation depends only on the probability of the event in question, the type of event, and the number of packets in the maximum congestion window. We illustrate the procedure by calculating the degradation for an example link.

When computing TCP throughput, there are two distinct cases: Depending on the bandwidth-delay product, throughput can be limited either by the bandwidth of the path across the network, or by the maximum TCP window size.

If the product of the end-to-end roundtrip delay and the available bandwidth is small, compared to the maximum TCP window size, spurious timeouts and spurious fast retransmit need not lead to any performance degradation. A modest buffer before the radio link will be enough to keep the link busy even when the sender temporarily decreases its sending rate. On the other hand, if the bandwidth-delay product is larger than the maximum TCP window size, throughput is decreased. The difference between these two cases can be seen for example in the performance evaluation [27]: In the scenarios that have a large maximum window size compared to the bandwidth–delay product, we get a throughput that is the nominal radio link bandwidth times $1 - p$ (where p is the average frame loss probability), and there is no significant difference between different link retransmission schemes. Only when bandwidth or delay is increased, or the maximum window size is decreased, do we see large changes in throughput when the frame error rate or retransmission-scheme varies.

Therefore, we will concentrate on the case of a large bandwidth-delay product. For a concrete example, consider the following scenario, illustrated in Figure 4.10:

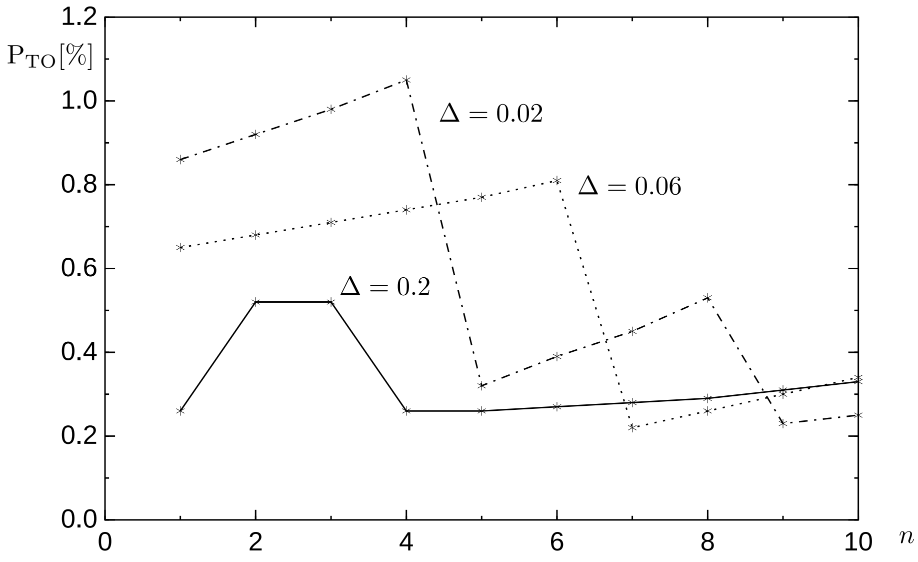

Figure 4.9: Probability of spurious timeout as a function of n, the number of radio frames per IP packet, for three different values of the power control step size Δ.

Radio link bandwidth 384 kbit/s, packet size $m = 1500$ bytes, maximum TCP window size $w = 5m = 7500$ bytes, and a constant roundtrip delay time, excluding the radio link itself, of 0.2 s. We also consider a smaller packet size of $m = 960$, with $w = 8m = 7680$ bytes, making each IP packet fit in a single radio frame. For simplicity, we always choose w as a multiple of m.

Let T denote the average end-to-end roundtrip time, 0.21 s for the larger packets and slightly less (0.206 s) for the smaller packets. This implies that the bandwidth-delay product is larger than the maximum window size, with an ideal throughput w/T of 34.8 Kbyte/s for the case of large packets and somewhat larger, 36.3 Kbyte/s, for the case of small packets. Note that both values are smaller than the available radio bandwidth of the link, $384000(1 - p_{\text{ref}})/8 \approx 42.2$ Kbyte/s.

Fast retransmit followed by fast recovery, and timeout followed by slow start, can be treated in a uniform way. Assume that fast retransmit (or timeout) occurs independently with a small probability P_{LOSS}. Consider a typical cycle starting with the event that causes the performance degradation, followed by $1/P_{\text{LOSS}}$ packets that are sent without timeouts or retransmissions. We want to compare the throughput during this cycle with the ideal throughput w/T. For simplicity, use only congestion windows that are an integral number of packets, and also assume that the cycle length $N = 1/P_{\text{LOSS}}$ is an integer. The cycle can be divided into two phases: An initial recovery phase of j roundtrip times, when ℓ packets are sent, followed by the second phase of $(N - \ell)m/w$ roundtrip times at full speed, w/m

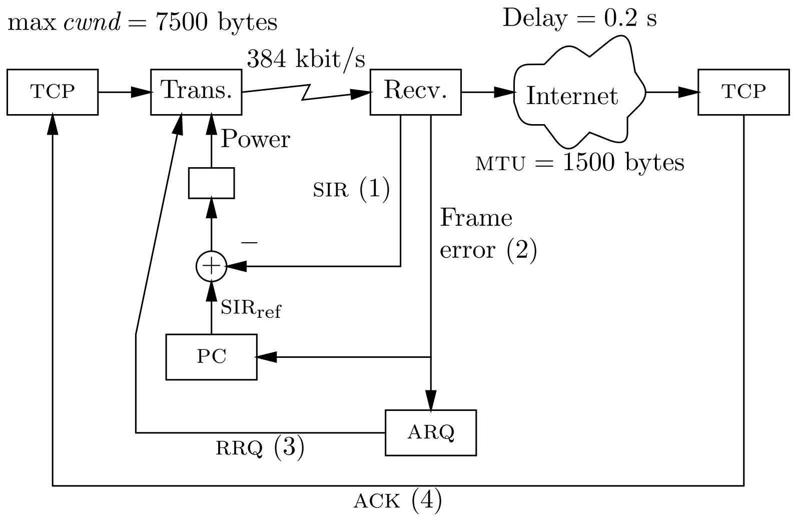

Figure 4.10: Numerical example.

packets each roundtrip time, $N - \ell$ packets in all.

The throughput during one cycle can be expressed as

$$\text{throughput} = \frac{N}{j + (N - \ell)m/w}\,\frac{m}{T}$$

Compared to the ideal throughput w/T, the relative degradation is

$$\left(\frac{w}{T} - \text{throughput}\right) \Big/ \frac{w}{T} = \frac{d}{d + N} = \frac{1}{1 + 1/(d\,\mathrm{P}_{\mathrm{LOSS}})}$$

where we have substituted $d = jw/m - \ell$, the number of additional packets that would have been sent during the cycle if TCP had never decreased the congestion window.

Degradation due to fast retransmit

First consider the effect of spurious fast retransmit and fast recovery. Here, the sending TCP will resend the delayed packet (thinking that it is lost), and halve its congestion window. The window size is then increased by m each RTT, until it reaches the maximum value, w.

For $m = 1500, w = 7500 = 5m$, the congestion window is halved to $w/2$ and, to keep the window size as an integral number of packets, we round this down to $2m$. During the next three roundtrip times TCP sends 2, 3, 4 packets, or 9 packets total, after which it is up to full speed again, sending 5 packets each roundtrip time. This means that $\ell = 9$, $j = 3$ and $d = 6$: Due to the spurious fast retransmit event,

Degradation	Packet size	$\Delta = 0.2$	0.06	0.02
due to P_{FR}	$m = 960$	2.3%	5.9%	7.3%
	1500	0.0%	0.1%	0.3%
due to P_{TO}	$m = 960$	5.6%	13.0%	16.5%
	1500	4.9%	6.4%	8.4%

Table 4.3: Performance degradation, due to spurious fast retransmit and spurious timeout.

6 packets less are sent during the cycle. The resulting performance degradation is $1/(1 + 1/(6\,P_{FR}))$. As each packet corresponds to two radio frames, $n = 2$ in Table 4.2. The performance degradation is noticeable only for the smaller step sizes. With $\Delta = 0.06, P_{FR} \approx 0.02\%$ yields $N = 5000$ and a performance degradation of $6/5006 \approx 0.1\%$, while $\Delta = 0.02, P_{FR} \approx 0.05$ yields $N = 2000$ and a performance degradation of $6/2006 \approx 0.3\%$.

For the packet size $m = 960$, the situation is different. The maximum window size is $w = 8m$, which is halved at spurious fast retransmit. During the 4 roundtrip times of the recovery phase, TCP sends 4, 5, 6 and 7 packets, giving $d = 10$. The large step size, $\Delta = 0.2$ yields $P_{FR} \approx 0.24\%$, $N = 417$, and a performance degradation of $10/427 \approx 2.3\%$. For the smaller step sizes $\Delta = 0.06$ and 0.02, the performance degradation is worse, 5.9% and 7.3%, respectively.

These figures are summarized in Table 4.3. The degradation is significant for the small packet size, and much less of a problem for larger packets.

Degradation due to timeouts

Next, consider the effect of timeout. When recovering from timeout, the slow start threshold is set to $w/2$. The congestion window is reset to one packet, and doubled each roundtrip time, until it reaches the slow start threshold. Above the slow start threshold, the usual increment of the congestion window of one packet per roundtrip time is used until we get back to the maximum value w.

For $m = 1500, w = 5m = 7500$, the length of the recovery phase is 4 roundtrip times, during which we send 1, 2, 3 and 4 packets. We get $d = 10$, and from the P_{TO} calculated in Section 4.4 we get a performance degradation of 4.9%, 6.4%, and 8.4% for the three step sizes.

For $m = 960, w = 8m = 7680$, the length of the recovery phase is 6 roundtrip times, during which we send 1, 2, 4, 5, 6, 7 packets. Thus, $d = 23$, and we get a performance degradation of 5.6%, 13.0% and 16.5% for the three step sizes.

These figures are also shown in Table 4.3. We see that degradation due to spurious timeout is significant for all considered values of the packet size and power control step size.

Influence of the power control step size

The choice for the step size Δ is determined by the tuning of the power control. It is is a tradeoff between short response times and energy efficiency, as discussed in Section 4.2. However, as can be seen in Table 4.3, the choice of Δ also influences TCP performance. For both spurious timeout and spurious fast retransmit, we see that the degradation gets worse as the power control step size Δ is made smaller.

To understand why, note that a small step size implies a low correlation in the frame loss process (in the limit $\Delta \to 0$, frame errors become independent). Then we have almost the same loss probability for original transmissions and for frame retransmissions. On the other hand, a large step size implies a significant increase in transmission power after each frame loss, which means that each retransmission of a frame will use a significantly higher SIR than the previous attempt. This reduces the number of retransmissions, which in turn reduces the retransmission delays and the problems with spurious fast retransmit and spurious timeout.

4.6 Summary

The properties of the lossy radio link considered in this chapter reduces TCP throughput. By modeling the underlying processes controlling transmission power and retransmission scheduling on the link, we can derive the link properties, in particular the delay distribution, that are relevant for TCP. We found that when using IP packets that fit in one or two radio frames, as is common for high-bandwidth wireless links, the link will trigger spurious fast retransmissions in TCP. Spurious timeouts will also occur, and this effect is significant for both large and small packets.

Chapter 5

Improving the link layer

This chapter proposes a measure of TCP*-friendliness, and we see how adding carefully chosen additional delays can make a link work better for* TCP*.*

5.1 Introduction

We believe that, as far as possible, the link-layer should be engineered to be TCP-friendly, reducing the differences between wired and wireless links. This chapter investigates ways to optimize the operation of link-layer processes, in order to improve TCP performance, and it consists of two mostly independent parts.

Section 5.2 proposes a model for analyzing and optimizing the retransmission scheduling. It is intuitively appealing to view retransmissions as a form of feedback, but non-trivial to find the right way to express this idea in a mathematical form.

The main part is Sections 5.3–5.5. These sections discuss modifications to the wireless link of the previous chapter, to reduce the TCP performance degradation due to spurious timeouts. The timeout procedure in TCP motivates a definition of a quality measure of TCP friendliness. The main result is that TCP performance can be improved by adding carefully chosen artificial delays to certain packets.

There will naturally be some residual idiosyncrasies of wireless channels that cannot be dealt with in the link-layer; the work presented in this chapter should be viewed as complementing both developments to make TCP more robust to “strange” links, and cross-layer developments that let the link and the end-nodes exchange information about link and flow properties.

5.2 Retransmission as feedback

To be able to analyze the impact of the scheduling mechanism on link properties such as the delay distribution, it is of interest to model the retransmission scheme. Feedback is an intrinsic property of the retransmission mechanism. Below we pro-

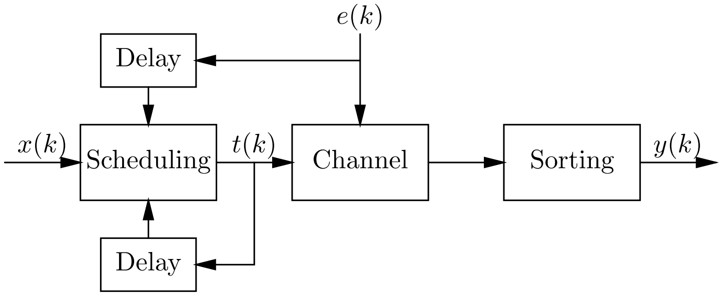

Figure 5.1: Retransmission model.

pose a model for the relationship between the input frames, the frame error process, and the in-order output frames, where this feedback is explicitly shown. We believe that this model will be useful for further studies of retransmission scheduling.

Let k denote time in units of the transmission time interval (TTI), and consider the following input and output signals, also shown in Figure 5.1.

$$\begin{aligned} x(k) &= \#\text{ of input frames up to time } k \\ e(k) &= \#\text{ of errors up to time } k \\ y(k) &= \#\text{ of in-order output frames up to time } k \end{aligned}$$

These are accumulated rate functions, also used in network calculus, and increasing. Also define

$$t(k) = \text{Index of frame transmitted at time } k$$

The function $t(k)$ is *not* increasing.

Consider a simple one-parameter family of retransmission schemes, where each damaged frame is retransmitted g slots later, and there is no limit on the number of times a frame may be resent. The parameter g corresponds to the RLP NAK guard time.

To describe the process mathematically, start with the queue at the input to the scheduler. Let $s(k)$ be the number of time slots up to time k that are not used for retransmissions, and let $f(k)$ be the number of frames that have been transmitted (but not necessarily received successfully) up to time k. Then

$$\begin{aligned} s(k) &= k - e(k-g) \\ f(k) &= \min_{\ell \le k} \left(x(\ell) + s(k) - s(\ell) \right) \end{aligned}$$

where the minimum in the latter equation is obtained when ℓ is the start of the current busy period.

The scheduling can be described as

$$t(k) = \begin{cases} f(k) & \text{if } e(k-g) = e(k-g-1) \\ t(k-g) & \text{if } e(k-g) > e(k-g-1) \end{cases} \tag{5.1}$$

Delay [ms]	0	40	60	100	120	160	180
Probability [%]	80.6	8.8	9.3	0.6	0.6	0.03	0.03

Table 5.1: Delay distribution of an example wireless link.

The first case corresponds to an original transmission, and the second case to a retransmission. Finally, $y(k)$ is defined by $y(k) = n$ if all frames up to n have been received properly at time k, but frame $n + 1$ has not. In symbols,

$$y(k) = \max\{n : \forall \ell \leq n, \exists j \leq k, t(j) = \ell, e(j) = e(j-1)\}$$

So where is the feedback? It is included explicitly, in (5.1). This model, together with a model for the stochastic process $e(k)$, lets us optimize the parameterized retransmission scheme. The delay at time k is defined by

$$d(k) = \min\{\tau \geq 0 : y(k + \tau) \geq x(k)\}$$

If x and e are stationary processes, with average rates that sum to less than one, then also $d(k)$ is a stationary process, and its properties can, at least in principle, be calculated from x, e, and the retransmission model. If $Q(d)$ is a quality measure that depends on the properties of d, one can formulate the optimization problem

$$g^* = \arg\max_g Q(d)$$

which gives the optimal value for the retransmission delay.

Intuitively, we expect that g^* will depend on the autocorrelation of e; it seems reasonable to use a retransmission delay such that the correlation, between loss of the original transmission and loss of the retransmission, is small.

This one parameter retransmission model is quite limited. Other schemes can be modeled analogously, as long as the relation between e and s is simple, and the scheme does not need an additional queue for retransmitted packets. The challenge is to find a powerful but simple parameterization of an interesting class of schemes.

We now leave the subject of retransmission scheduling, and develop a particular quality measure.

5.3 IP packet delay

Since a link employing link-layer retransmission yields a very small packet loss probability, the most important characteristic of the link is the packet delay distribution. If the distribution is sufficiently "friendly" to TCP, then the layering of the system works nicely, which means that upper layers like TCP need not be aware of any particular properties of individual links in the network.

In the previous chapter, we computed the packet delay distribution explicitly, for a channel with SIR-based power control. The resulting delay distribution for our example channel, with a power control step size $\Delta = 0.06$ dB and $n = 2$ is shown in Table 5.1. This table includes only the delays due to radio frame retransmissions, there is also a fixed delay of 40 ms for the transmission of two radio frames.

TCP performance degradation

In observations and performance evaluations of TCP over wireless links [27], the properties of a wireless link can shine through to the TCP layer in three different ways:

Genuine packet loss: With bad enough radio conditions, packet drops are inevitable. We will not consider such genuine packet loss here, as we assume that the radio channel is good enough that the power control and link-layer transmissions can get packets through.

Packet reorder: For a link with highly variable delay, packets can get reordered. Severe reordering can trigger a spurious TCP fast retransmit.

Spurious timeout: A packet that is not lost, only severely delayed, can trigger a spurious TCP timeout.

Spurious timeout

A TCP timeout event occurs when a packet, or its acknowledgment, is delayed too long, as determined by the timeout value RTO. RTO is computed from estimates of the mean and variance of the RTT, as described in the previous chapter.

An idealized model of TCP is to assume that the estimation is perfect, so that the timeout value is set as

$$\text{RTO} = \mu(\text{RTT}) + 4\sigma(\text{RTT})$$

where $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation. Then the spurious timeout probability is given by

$$\mathrm{P}_{\mathrm{TO}}(\text{RTT}) = \mathrm{P}(\text{RTT} > \mu(\text{RTT}) + 4\sigma(\text{RTT})) \tag{5.2}$$

Note that P_{TO} is invariant under addition of constant delays.

For the delay distribution of Table 5.1, we get RTO $\approx$ 103 ms and the probability that the delay is larger is $\mathrm{P}_{\mathrm{TO}} \approx 0.68\%$. In the previous chapter, we had a spurious timeout probability on the order of 0.5%–1%, when varying the packet size n and power control step size Δ, and we saw that even a fairly small P_{TO} can result in significant performance degradation.

5.4 Improving the link-layer

It is not trivial to define precisely what properties a link should have in order to be friendly to TCP. It seems clear that for example links with normal or uniformly distributed and independent delays are friendly enough. We define a measure of TCP-friendliness by applying (5.2) to an arbitrary stochastic variable X, representing the independent and identically distributed packet delays. We also define $\text{RTO}(X)$ as the corresponding timeout value.

$$\begin{aligned} \text{RTO}(X) &= \mu(X) + 4\sigma(X) \\ \text{P}_{\text{TO}}(X) &= \text{P}(X > \text{RTO}(X)) \end{aligned}$$

Let us compare the P_{TO} for the wireless delay of Table 5.1, 0.68%, to the P_{TO} of some other distributions:

- If X is uniformly distributed on an interval $a \leq X \leq b$, then it has mean $\mu = (a+b)/2$, and standard deviation $\sigma = (b-a)/(2\sqrt{3})$, so that $\mu + 4\sigma = b + (2/\sqrt{3} - 1/2)(b-a) > b$. Hence, $\text{P}_{\text{TO}}(X) = 0$.

- If X is Gaussian, then timeouts will be rare: $\text{P}_{\text{TO}}(X) = \frac{1}{\sqrt{2\pi}} \int_4^\infty e^{-x^2/2}\,\text{d}x \approx 0.003\%$.

- For an arbitrary distribution with finite mean and variance, Chebyshev's inequality, $\text{P}(|X-\mu| \geq a\sigma) \leq 1/a^2$, yields the bound $\text{P}_{\text{TO}} \leq 6.25\%$.

We see that the first two distributions, which we know are friendly to TCP, yield a P_{TO} at least two orders of magnitude below the worst case given by Chebyshev. The wireless delay yields a significantly higher P_{TO}, although still with some margin to the worst case.

The motivation for using P_{TO} as a quality measure is the calculation of the timeout value in TCP. Timeout is intended to be the last resort recovery mechanism, and for TCP to work properly, spurious timeout must be a rare event. We therefore define a TCP-*friendly link* to mean a link with no loss or reorder, and with a delay distribution that yields a small P_{TO}.

It would be nice to use the P_{TO} measure to optimize the retransmission scheduling, as described in Section 5.2. But in the remainder of this chapter, we will follow a much simpler route, which never the less allows us to decrease the P_{TO} considerably: Addition of artificial delays.

Introducing additional delays

Assume that we have a discrete delay distribution X, $\text{P}(X = d_i) = p_i$, where $d_i < d_{i+1}$. It is typical, but not required, that also $p_i \geq p_{i+1}$.

We consider the following class of tweaks to X. For each packet that experiences a delay $X = d_i$, buffer the packet so that it gets an additional delay δ_i. This defines a new distribution $\widetilde{X}$, $\text{P}(\widetilde{X} = d_i + \delta_i) = p_i$ (or if it happens that $d_i + \delta_i = d_j + \delta_j$ for some $i \neq j$, the corresponding probabilities are added up). For an example of what X and $\widetilde{X}$ can look like, see Figures 5.2.

What is the best choice for $\delta_i \geq 0$? One possible answer is given by the optimization problem

$$\min_{\delta_i \geq 0} \mu(\widetilde{X})$$
$$\mathrm{P}_{\mathrm{TO}}(\widetilde{X}) \leq \epsilon$$

where $\epsilon > 0$ is a maximum allowed value for $\mathrm{P}_{\mathrm{TO}}(\widetilde{X})$. This means that we want to push down our measure of TCP-unfriendliness, while at the same time not adding more delay than necessary. We will see that after some simplifications, this is a quadratic optimization problem.

First, require that $\mathrm{P}_{\mathrm{TO}}(\widetilde{X})$ corresponds to a tail of the original distribution X. Let k be the smallest value such that $\sum_{i \geq k+2} p_i \leq \epsilon$. Let $c = d_{k+1} + \rho < d_{k+2}$, where $\rho \geq 0$ is a robustness margin. We impose the additional constraints $d_i + \delta_i \leq d_{k+1}$ for $i \leq k$, $\delta_i = 0$ for $i > k$, and $\mathrm{RTO}(\widetilde{X}) = c$. Then, for any δ_i satisfying these new constraints, we will have $\mathrm{P}_{\mathrm{TO}}(\widetilde{X}) = \sum_{i \geq k+2} p_i \leq \epsilon$. We get the optimization problem

$$\min_{\delta_1, \dots, \delta_k} \mu(\widetilde{X})$$
$$\mu(\widetilde{X}) + 4\sigma(\widetilde{X}) = c$$
$$0 \leq \delta_i \leq d_{k+1} - d_i, \text{ for } i \leq k$$

To write it in matrix form, let δ denote the vector $(\delta_1, \dots, \delta_k)^T$, and similarly for p and d. Let $S = 16 \operatorname{diag} p - 17pp^T, b_i = 2p_i(16d_i + c - 17\mu), m_i = d_{k+1} - d_i$ and $\alpha = 16\sigma^2 - (c - \mu)^2$, where μ and σ denote the mean and standard deviation of the original delay X. We can then rewrite the problem as

$$\min_{\delta_1, \dots, \delta_k} p^T \delta$$
$$\delta^T S \delta + b^T \delta + \alpha = 0$$
$$0 \leq \delta_i \leq m_i$$

Remarks

Since the symmetric matrix S is typically indefinite, the problem is not convex. But it can be solved in exponential time $O(k^3 3^k)$, which has not been a problem thanks to the very limited size of k.

The rôle of ρ can be seen in "after" graph in Figure 5.2; it is the margin, on the delay axis, between the RTO $= \mu + 4\sigma$ and the delay value closest to the left (120 ms).

The typical solution is of the form $x = (0, \dots, 0, \delta_j, m_{j+1}, \dots, m_k)^T$. When the optimum has this form, it means that the cheapest way to increase the RTO, in terms of mean delay, is to increase the δ_i corresponding to the smallest p_i. Necessary and sufficient conditions for the optimum to be of this form has not yet been determined.

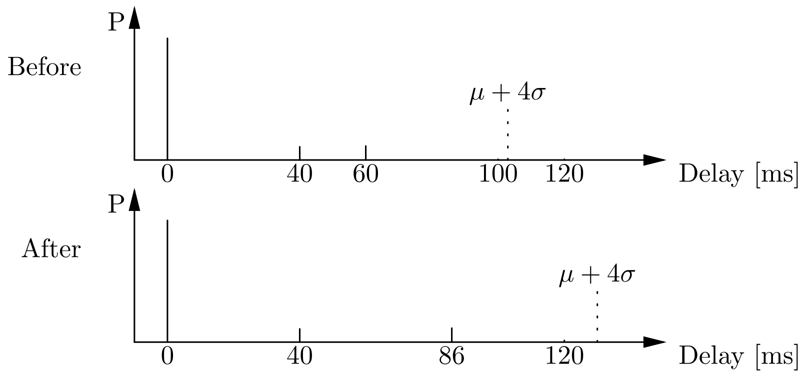

Figure 5.2: Delay distribution, before and after optimization.

We aim at decreasing the distribution tail as measured by P_{TO}, and pay a small price in mean delay. This is a different objective than an ordinary jitter buffer, which aims for small variance, and pays with a larger mean delay.

Numerical example

Now consider the delays d_i and probabilities p_i in Table 5.1, and assume that packets are independently delayed according to the given probabilities. This distribution is also shown at the top of Figure 5.2. Before tweaking the delays, we have $E(X) \approx$ 10.6 ms, and $P_{TO}(X) \approx 0.68$ %.

With $\epsilon = 0.1\%$ and $\rho = 10$ ms, the above optimization procedure yields $k = 4$ and the optimal additional delay $\delta \approx (0, 0, 26, 20)^T$ ms. This modified distribution is shown at the bottom of Figure 5.2. The mean additional delay is only 2.54 ms, which seems to be a small cost, if we compare it to the propagation delay for the packet, which is 40 ms, or the end-to-end delay which necessarily is even larger. We also achieve $P_{TO} < \epsilon$, if fact, we actually get $P_{TO} \approx 0.06\%$.

Consider the same example scenario as in the previous chapter, illustrated in Figure 4.10. The available radio bandwidth, 384 kbit/s with 10% losses, corresponds to 42.2 Kbyte/s. The ideal TCP performance, one maximum window per RTT, was 34.8 Kbyte/s, which was reduced by 6.4% to 32.6 Kbyte/s due to spurious timeout events.

For the tweaked link, we have a slightly larger RTT, which in itself would decrease the throughput by 1%, and a significantly smaller P_{TO}. The resulting throughput is 34.2 Kbyte/s, an improvement by 5% compared to the unmodified link, and only 1% below the ideal TCP throughput. The degradation due to spurious timeout is almost eliminated. These figures are summarized in Table 5.2.

The important point is that a simple but carefully selected modification to the link-layer yields a modest but significant performance improvement.

	Kbyte/s
Available radio bandwidth	42.2
Ideal TCP throughput	34.8
With wireless link	32.6
With optimized wireless link	34.2

Table 5.2: The TCP throughput for the wireless link, before and after optimization.

5.5 Robustness

In the analysis, we have so far only considered the delay variation from the wireless link, and assumed that the delay over the rest of the network path is constant. Now assume that the artificial delays are chosen optimally based on the delay distribution for the isolated link, and applied to the end-to-end RTT over the complete network path. In this section we will prove that the resulting P_{TO} is still small, under some conditions on the delay distribution for the rest of the network.

If the delay in the rest of the network is modelled as a stochastic variable V, then the end-to-end RTT is $\widetilde{X} + V$. So our objective is to find a bound on $\mathrm{P}_{\mathrm{TO}}(\widetilde{X} + V)$. Since P_{TO} is invariant under the addition of constant delays, we can make the convenient assumption that $\mathrm{E}(V) = 0$.

If we assume that $\widetilde{X}$ and V are independent, we can derive a simple bound for $\mathrm{P}_{\mathrm{TO}}(\widetilde{X} + V)$, which will also depend on the robustness margin ρ. First, define $c' = \mathrm{E}(\widetilde{X} + V) + 4\sigma(\widetilde{X} + V)$. Note that $c' \geq c$, so that $c' - x_i - d_i \geq \rho$ for $i \leq k$. Next, condition on $\widetilde{X}$,

$$\begin{aligned}
\mathrm{P}_{\mathrm{TO}}(\widetilde{X} + V) &= \mathrm{P}(\widetilde{X} + V > c') \\
&= \sum_i p_i \, \mathrm{P}(V > c' - x_i - d_i) \\
&\leq \sum_{i=1}^{k} p_i \, \mathrm{P}(V > \rho) + \sum_{i>k} p_i \\
&\leq \mathrm{P}(V > \rho) + \epsilon
\end{aligned}$$

This bound shows that if the delay variations in the rest of the network are small enough relative to our robustness parameter ρ, $\mathrm{P}_{\mathrm{TO}}(\widetilde{X} + V)$ will not be much larger than ϵ. For typical distributions, the bound for the first sum is very conservative. This is because the first few p_i dominates, while the corresponding $c' - x_i - d_i$ are significantly larger than ρ. A more precise bound can be calculated using additional information about p_i and the distribution of V.

5.6 Summary

The first part of this chapter discusses optimization of the retransmission scheduling. An input/output model is suggested where the rôle of the scheduling mechanism is made explicit as a feedback.

The main contribution has been to show that a slight artificial increase of the delays of certain retransmitted packets may reduce the risk of spurious timeout in TCP and hence increase the throughput; in an example the increase was 5%. The artificial delay distribution is optimized off-line and applied on-line. The additional delay that is applied to a packet depends only on the retransmission delay experienced by that same packet, and this information is available locally at the link.

Chapter 6

Conclusions and further work

This chapter summarizes the results of the preceding three chapters, and outlines directions for future research.

6.1 Radio network feedback

Chapter 3 studies TCP performance over the high-speed downlink packet access (HSDPA) channel in the universal mobile telecommunications system (UMTS), i.e., downlink performance. The channel disturbances in our model and simulations are bandwidth changes and temporary outages. A split-connection scheme is proposed, analyzed, and compared to a setup with standard TCP Reno end-to-end.

The proposed scheme uses an HTTP proxy, and explicit signalling from the radio network controller (RNC) to the proxy. The proxy does not use TCP Reno to determine its window size and sending rate, instead it uses a controller based on feedforward of the actual bandwidth of the radio channel, and feedback from the queue length at the RNC. The information needed by the controller is provided by the RNC in the form of radio network feedback (RNF) messages.

With this proxy, the user's response time is shortened, and the utilization of the radio channel is improved, compared to TCP Reno. Thanks to the feedforward control, the proxy can adapt faster to bandwidth changes. In particular, it recovers better from temporary outages. The proxy gets accurate RNF messages at the start and end of an outage. In contrast, TCP Reno, like any end-to-end protocol, will see no ACKs at all during an outage, and it is required to use exponential backoff when probing the channel to see when the outage ends.

The model based design allows tuning of parameters such as the target RNC queue size q_{ref}, and the amount of buffering space for the queue. The proposed controller uses well known principles: Feedforward, to adjust to bandwidth changes, and feedback, to control the RNC queue size in spite of uncertainties in the available bandwidth and delay information. The actual control laws are very simple; more advanced controllers could be designed to improve performance, or to handle additional disturbances in the system. One interesting extension is to introduce an outer control loop that selects a suitable value for q_{ref}.

The stability analysis in Chapter 3 is limited, it would be desirable to improve the analysis to prove global stability of the non-linear system, including the propagation delays.

Implementation of the proxy and the RNF-enabled RNC is needed, to evaluate the performance of the proxy solution for the real system, and to compare it to other alternatives, e.g., performance enhancing proxies.

6.2 Delay variations of wireless links

Since modern wireless links include the capability of local retransmissions of damaged frames, the primary difference between such a link and a wired one is no longer the loss rate, but the packet delay distribution.

Chapter 4 develops a model for the frame loss process of a wireless link, using models for the power control and retransmission processes. This model lets us derive the link properties that are relevant to TCP, in particular the delay distribution.

To quantify the impact on TCP, the performance degradation due to spurious timeout and spurious fast transmit was computed, and found to be up to 15%. A simple way to alleviate the performance degradation caused by spurious fast retransmissions is to configure the link to not forward packets out of order. It seems clear that for links where TCP performance is important, and an option of in-order delivery is available, it should be enabled.

An accurate model for the link-layer processes, and the resulting delay distribution, makes it possible to tune system parameters in a systematic way. The model can also be used for the design of new controllers. A first step in this direction is described in Chapter 5, which addresses the problem of spurious timeout. We define a measure of TCP-friendliness for a link, or more precisely, for the link's delay distribution. The timeout value in TCP makes timeout a very rare event if the roundtrip time has a uniform or normal distribution, but that is not the case for the delay due to link-layer retransmissions.

A wireless link can be made more friendly to TCP, in this sense, by adding carefully chosen artificial delays to certain packets. Choosing these additional delays, and at the same time keep the average delay increase small, is posed as an optimization problem. If the link adds the optimal delay to each packet, the average delay increase is only a few ms, but still most of the spurious timeouts, and the corresponding performance degradation of TCP, is eliminated.

One interesting mathematical problem is extending the delay optimization to a continuous setting. One variation is as follows: Consider an input which is a stochastic process $x(t)$, a non-negative control signal $d(t)$, and the output $\widetilde{x}(t) = x(t) + d(t)$. Choose an optimal $d(t)$, depending on the history of $x(t)$, subject to an objective and a set of constraints, which are all expressed as functionals of $\widetilde{x}(t)$.

The used models need further experimental validation. The results are consis-

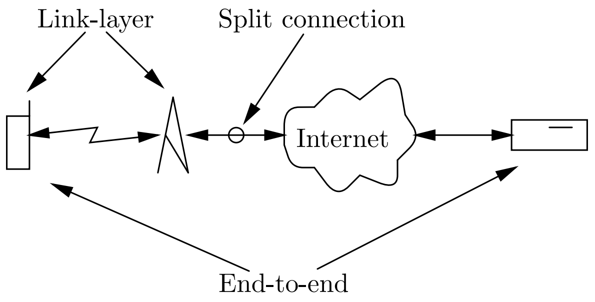

Figure 6.1: Approaches to the TCP over wireless problem.

tent with experiments and simulations in the literature, which focus on high level properties such as TCP throughput. Data sets with realizations of the underlying frame loss process would be highly useful for validation.

6.3 Approaches to TCP over wireless

As described in the introduction and illustrated in Figure 6.1, there are several possible approaches to improved TCP performance over radio links, classified as:

End-to-end: Improve the transport algorithms in the endpoints, i.e., TCP, to adapt better to the disturbances from wireless links.

Split-connection: Use a proxy to hide the wireless link from peers on the Internet.

Link-layer: Design link-layer tradeoffs that results in link properties that are easier to handle for the endpoints.

Improving the end-to-end TCP algorithms is an area of intense research. A drawback of this approach is that deployment of new algorithms affect all Internet end systems, which is a slow and costly process. Tuning the link properties is more practical from a deployment point of view, at least if the tuning can be done before widespread adoption of a new link type. For example, spurious fast retransmissions can be eliminated by having the wireless receiver buffer packets, making sure that packets are never forwarded out of order (this “in-order” option is already available on real systems [13, 32]). To reduce the number of spurious timeouts, the link can add artificial delays to packets or acknowledgements, to make the delay variation more friendly to TCP, either random delays as in [28], or as described in Chapter 5. Also the link-layer retransmissions themselves can be seen as an example of changing the link properties to make the link more TCP friendly.

To this list, one could also add overall delay reduction. Smaller delays improves network performance in general and TCP performance in particular. If the bandwidth-delay product can be made smaller than the maximum window size in

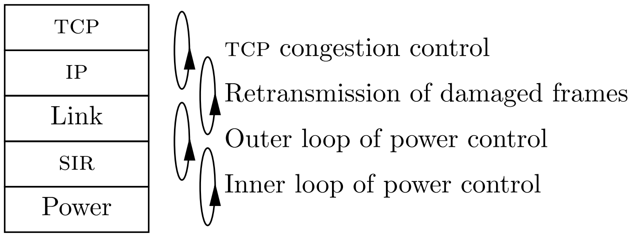
TCP
IP
Link
SIR
Power
TCP congestion control
Retransmission of damaged frames
Outer loop of power control
Inner loop of power control

Figure 6.2: The layers in the networking stack, and the corresponding control loops.

TCP, then a modest size buffer just before the radio link should be sufficient to keep the link saturated, regardless of fluctuations in the TCP congestion window. It is hard to say if reducing bandwidth-delay product is a realistic objective, as it depends on the speed and delay of future link technologies. Extensions to TCP that increase the maximum window size would also help, for the same reasons [25].

6.4 Towards a systematic design for layer separation

If we look at the networking stack in Figure 6.2, the ideal is to have *layer separation*: Each layer should operate independently and not interact with the inner workings of other layers. In control theory, a similar notion is *cascaded control*, where each loop operates on its own time-scale, each inner loop working on a faster time-scale than the outer loops. In each layer, there are tradeoffs that are influenced by the design of the control loops. If we want to improve layer separation by changing the automatic control in the various layers, where should we put the effort?

In the lowest layer, the power control, the control design has severe constraints of its own, relating to energy efficiency, stability and cost of deployment. We can probably not add new requirements, related to "TCP-friendliness", to power control design.

At the opposite end of the networking stack, end-to-end congestion control, improvements to the TCP algorithms in the end-nodes are important, but difficult for several reasons. Since TCP has to be general and work over any path across the heterogeneous Internet, with many different types of links, it does not make sense to tailor TCP for the peculiarities of one link type. End nodes have limited information about what goes on in the link (note that the link and the TCP implementations are not only in separate layers, they are also *geographically* separate). There are also severe design constraints, such as fairness and global stability of the Internet.

However, we do have a fair amount of engineering freedom in the link-layer, even if we do not want to modify the power control. In this thesis we have considered only one out of many possible schemes for link-layer retransmissions (ARQ). There are also other link-layer mechanisms that can be exploited, e.g., forward error correction (FEC), rate adaptation and scheduling. Joint optimization of FEC, ARQ and transmission power is investigated in [6], and extended further, to also include rate adaptation, in [38].

How to get the most out of these different mechanisms, used together, is not well understood. The output signals we want to control are the capacity, loss and delay. It is clear that the available control signals enables us to make tradeoffs between how radio disturbances affect the output signals. But how should the disturbance rejection work be divided between power control, rate adaptation, FEC, ARQ, and scheduling? What are the fundamental limitations on what can be achieved?

To achieve a general layer separation between TCP and radio links, we need a class of "friendly" link characteristics, in terms of the dynamics of capacity, losses and delay, such that:

- Link-layer control can be used to turn real physical radio channels into links of this class, with close to full utilization of the radio resources.
- Congestion control of end-to-end TCP can be designed to work well over all links in the class, and over all network paths made up of links from this class.

Searching for such a class, and designing the corresponding link-layer and end-to-end controllers, is a meet-in-the-middle attack on the TCP-over-wireless problem.

Bibliography

[1] A. A. Abouzeid, S. Roy, and M. Azizoglu. Stochastic modeling of TCP over lossy links. In *INFOCOM (3)*, volume 3, pages 1724–1733, 2000.

[2] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999.

[3] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *SIGCOMM*, Portland, September 2004. ASM.

[4] Y. Bai, P. Zhu, A. Rudrapatna, and A. T. Ogielski. Performance of TCP/IP over IS-2000 based CDMA radio links. In *IEEE VTC'2000-Fall*. IEEE, 2000.

[5] C. Barakat and E. Altman. Bandwidth tradeoff between TCP and link-level FEC. *Comput. Networks*, 39(5):133–150, 2002.

[6] D. Barman, I. Matta, E. Altman, and R. El Azouzi. TCP optimization through FEQ, ARQ and transmission power tradeoff. In *International Conference on Wired/Wireless Internet Communications WWIC*, february 2004.

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

[8] A. Canton and T. Chahed. End-to-end reliability in UMTS: TCP over ARQ. In *IEEE Globecom*, 2001.

[9] G. Carneiro, J. Ruela, and M. Ricardo. Cross-layer design in 4G wireless terminals. *IEEE Wireless Communications*, 11(2):7–13, April 2004.

[10] S. Cen, P. C. Cosman, and G. M. Voelker. End-to-end differentiation of congestion and wireless losses. *IEEE/ACM Trans. on Networking*, 11(5):703–717, 2003.

[11] T. Chahed, A-F. Canton, and S-E. Elayoubi. End-to-end TCP performance in W-CDMA/UMTS. In *ICC*, Anchorage, May 2003.

[12] A. Chockalingam, A. Zorzi, and V. Tralli. Wireless TCP performance with link layer FEC/ARQ. In *IEEE ICC*, pages 1212–1216, June 1999.

[13] A. Dahlén and P. Ernström. TCP over UMTS. In *Radiovetenskap och Kommunikation 02*, RVK, 2002.

[14] A. DeSimone, M. Chuah, and O. Yue. Throughput performance of transport-layer protocols over wireless LANs. In *IEEE Globecom'93*, pages 542–549, 1993.

[15] J. Postel (ed.). Transmission control protocol. RFC 793, September 1981. DARPA Internet Program.

[16] R. Braden (ed.). Requirements for internet hosts — communication layers. RFC 1122, October 1989.

[17] G. Fairhurst and L. Wood. Advice to link designers on link automatic repeat request (ARQ). RFC 3366, August 2002.

[18] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, April 1999.

[19] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An extension to the selective acknowledgement (SACK) option for TCP. RFC 2883, July 2000.

[20] C. P. Fu and S. C. Liew. TCP veno: TCP enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications*, 21(2):216–228, 2003.

[21] F. Gunnarsson and F. Gustafsson. Power control in wireless communications networks — from a control theory perspective. In *IFAC World Congress*, Barcelona, 2002.

[22] E. Hossain, D. I. Kim, and V. K. Bhargava. Analysis of TCP performance under joint rate and power adaptation in cellular WCDMA networks. *IEEE Trans. on Wireless Comm.*, 3(3), May 2004.

[23] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280(5360):95–97, 1998.

[24] V. Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18:314–329, 1988.

[25] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, May 1992.

[26] V. Kawadia and P. R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wireless Communications*, 12(1):3–11, Februari 2005.

[27] F. Khan, S. Kumar, K. Medepalli, and S. Nanda. TCP performance over CDMA2000 RLP. In *IEEE VTC'2000-Spring*, pages 41–45, 2000.

[28] T. E. Klein, K. K. Leung, R. Parkinson, and L. G. Samuel. Avoiding TCP timeouts in wireless networks by delay injection. In *IEEE Globecom*, 2004.

[29] R. Ludwig and R. H. Katz. The Eifel algorithm: Making TCP robust against spurious retransmissions. *ACM Computer Communication Review*, 30(1), January 2000.

[30] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang. TCP Westwood: bandwidth estimation for enhanced transport over wireless links. In *MobiCom*, Rome, Italy, 2001.

[31] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. RFC 2018, October 1996.

[32] 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.

[33] I. Cabrera Molero. Radio network feedback to improve TCP utilization over wireless links. Master's thesis, KTH, 2005.

[34] A. Mukherjee. On the dynamics and significance of low frequency components of internet load. Technical report, Univerity of Pennsylvania, 1992.

[35] C. Hava Muntean, J. McManis, and J. Murphy. The influence of Web page images on the performance of Web servers. *Lecture Notes in Computer Science*, 2093, 2001.

[36] J. Pan, J.W. Mark, and X. Shen. TCP performance and behaviors with local retransmissions. *Journal of Super Computing (special issue on Wireless Internet: Protocols and Applications)*, 23(3):225–244, 2002.

[37] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, September 2001.

[38] C. Rinaldi. Link-layer error recovery techniques to improve TCP performance over wireless links. Master's thesis, KTH, 2005.

[39] N. K. G. Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. *IEEE Proceedings-Communications*, 146(4):222–230, 1999.

[40] A. Sampath, P. S. Kumar, and J. M. Holtzman. On setting reverse link target SIR in a CDMA system. In *IEEE Vehicular technology conference*, May 1997.

[41] G. Xylomenos and G. C. Polyzos. Internet protocol performance over networks with wireless links. *IEEE Network*, 13(4):55–63, 1999.

Appendix A

Proof of Lemma 1

Lemma 1 *The equation $\alpha x = e^x$, with $\alpha > 0$, has a solution $x > 0$ if and only if $\alpha \geq e$. Furthermore, the solutions satisfy* $1/(\alpha - 1) < x < 2\alpha$.

Proof: For the first part, define $f(x) = -\alpha x + e^x$. We have $f(0) = 1$ and $f(x) \to +\infty$ as $x \to +\infty$. By continuity, f has a zero if and only if $\min f(x) \leq 0$. The minimal value can be computed explicitly: $f'(x) = -\alpha + e^x$. If $\alpha \leq 1$, f is strictly increasing and hence positive for all $x \geq 0$. So assume $\alpha > 1$. Then $f'(x) = 0$ has a single solution at $x = \log \alpha$. Which implies $\min f(x) = \alpha(-\log \alpha + 1) \leq 0$ if and only if $\alpha \geq e$.

For the second part, the inequality

$$\alpha x = e^x > 1 + x + \frac{1}{2}x^2$$

implies that $0 > x^2 - 2(\alpha - 1)x + 2 = (x - (\alpha - 1))^2 + 2 - (\alpha - 1)^2$. Since $\alpha \geq e$, we have $(\alpha - 1)^2 > 2$. Hence any solution x must lie in the interval

$$|x - (\alpha - 1)| < \sqrt{(\alpha - 1)^2 - 2}$$

For the lower bound, use $1 - \sqrt{1 - t} \geq t/2$, to get

$$\begin{aligned} x &> \alpha - 1 - \sqrt{(\alpha - 1)^2 - 2} \\ &= (\alpha - 1)\left(1 - \sqrt{1 - \frac{2}{(\alpha - 1)^2}}\right) \\ &\geq \frac{1}{\alpha - 1} \end{aligned}$$

and for the upper bound, $x < \alpha - 1 + \sqrt{(\alpha - 1)^2 - 2} < 2\alpha$. $\square$