Window-based congestion control

Modeling, analysis and design

NIELS MÖLLER

Doctoral Thesis
Stockholm, Sweden, 2008

TRITA-EE 2008:001
ISSN-1653-5146
ISBN-978-91-7178-831-3

KTH
SE-100 44 Stockholm
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägges till offentlig granskning för avläggande av teknologie doktorsexamen i telekommunikation 2008-01-25, 10:15 i Q1.

Tryck: Universitetsservice US AB

Abstract

This thesis presents a model for the ACK-clock inner loop, common to virtually all Internet congestion control protocols, and analyzes the stability properties of this inner loop, as well as the stability and fairness properties of several window update mechanisms built on top of the ACK-clock. Aided by the model for the inner-loop, two new congestion control mechanisms are constructed, for wired and wireless networks.

Internet traffic can be divided into two main types: TCP traffic and real-time traffic. Sending rates for TCP traffic, e.g., file-sharing, uses window-based congestion control, and adjust continuously to the network load. The sending rates for real-time traffic, e.g., voice over IP, are mostly independent of the network load. The current version of the Transmission Control Protocol (TCP) results in large queueing delays at bottlenecks, and poor quality for real-time applications that share a bottleneck link with TCP.

The first contribution is a new model for the dynamic relationship between window sizes, sending rates, and queue sizes. This system, with window sizes as inputs, and queue sizes as outputs, is the inner loop at the core of window-based congestion control. The new model unifies two models that have been widely used in the literature. The dynamics of this system, including the static gain and the time constant, depend on the amount of cross traffic which is not subject to congestion control. The model is validated using `ns2` simulations, and it is shown that the system is stable. For moderate cross traffic, the system convergence time is a couple of roundtrip times.

When introducing a new congestion control protocol, one important question is how flows using different protocols share resources. The second contribution is an analysis of the fairness when a flow using TCP Westwood+ is introduced in a network that is also used by a TCP New Reno flow. It is shown that the sharing of capacity depends on the buffer size at the bottleneck link. With a buffer size matching the bandwidth-delay product, both flows get equal shares. If the buffer size is smaller, Westwood+ gets a larger share. In the limit of zero buffering, it gets all the capacity. If the buffer size is larger, New Reno gets a larger share. In the limit of very large buffers, it gets 3/4 of the capacity.

The third contribution is a new congestion control mechanism, maintaining small queues. The overall control structure is similar to the combination of TCP with Active Queue Management (AQM) and explicit congestion notification, where routers mark some packets according to a probability which depends on the queue size. The key ideas are to take advantage of the stability of the inner loop, and to use control laws for setting and reacting to packet marks that result in more frequent feedback than with AQM. Stability analysis for the single flow, single bottleneck topology gives a simple stability condition, which can be used to guide tuning. Simulations, both of the fluid-flow differential equations, and in the `ns2` packet simulator, show that the protocol maintains small queues. The simulations also indicate that tuning, using a single control parameter per link, is fairly easy.

The final contribution is a split-connection scheme for downloads to a mobile terminal. A wireless mobile terminal requests a file from a web server, via a proxy. During the file transfer, the Radio Network Controller (RNC) informs the proxy 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 window size. In simulation studies, including one based on detailed radio-layer simulations, both the user response time and the link utilization are improved, compared TCP New Reno, Eifel and Snoop, both for a dedicated channel, and for the shared channel in High-Speed Downlink Packet Access.

Preface

He does love his numbers
And they run, they run, they run him
In a great big circle
In a circle of infinity

3.1415926535 897932
3846 264 338 3279

Kate Bush, π

Graduate education does not quite follow the general rule that the last 10% of any work takes 90% of the time. Of the results in this thesis, most were developed in the last two years, after my licentiate degree. For a long time, I thought that it would be best to stay outside of the area of congestion control, since it is so crowded. Searching the science citation index for "Congestion control" gives 333 journal articles published in the period 2006–2007 (excluding December 2007, which is still pending as I write this).[1]

During my first years as a PhD student, my main research interest was TCP over wireless. This is also an area with a lot of people working, but it is also wide area, with many different and loosely related problems. The main topic in my licentiate thesis, interactions between link-layer control, e.g., power control, and end-to-end congestion control, was a reasonably small and comfortable niche.

But in the end, I could not resist the temptation to study congestion control for the Internet. Like computer science, which is the study of man-made general purpose computers such as the von Neumann architecture and the Turing machine, networking research studies the Internet. This is another man-made machine constructed in the last decades, which is still growing, both in the number of users[2], and in its influence on everyday life. And congestion control is the single most important mechanism that prevents this machine from breaking down.

[1] And if conference papers, technical reports, and the like are included, the numbers are of course higher. With the same search criteria, `scholar.google.com` finds 4730 papers.

[2] Last month, the One Laptop per Child project finally started the mass production of its XO laptops, for distribution to school children in developing countries.

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

The material is the result of several collaborations: Within the control group and other groups at KTH. With Chadi Barakat, Konstantin Avrachenkov and Eitan Altman, at INRIA in Sophia Antipolis, where I spent couple of months as a guest during the spring of 2006. With Åke Arvidsson, Justus Petersson, Robert Skog, and Patrik Karlsson at Ericsson, where our project on Radio Network Feedback has been ongoing for several years. Of all the people in my and neighboring research groups, I would in particular like to thank those who have spent time and effort reviewing different parts of the thesis manuscript: My advisor Karl Henrik Johansson and my co-advisor Håkan Hjalmarsson, Carlo Fischione, who is currently at Berkeley but is expected back to KTH in coming months, Krister Jacobsson, Björn Johansson, Ulf Jönsson, and Camilla Trané.

The few photographs in the thesis are provided by my friends Jonas Ahrentorp and Kjell Enblom, and by the `www.flickr.com` user Todd Klassy, who made his photographs available under the Creative Commons attribution license. Their original color photos are more beautiful than the cropped and grey scale versions printed here.

Finally, thanks to Vilgot, who was not allowed to help me with the writing as much as he wanted to `fgbcvxåopl nbbbbbbbred3-.p ----nmgS,m m c kh fv c`.

Contents

Abbreviations

3GPP	3rd Generation Partnership Project
ACK	Acknowledgement
ADSL	Asymmetrical Digital Subscriber Line
AIMD	Additive Increase, Multiplicative Decrease
AMC	Adaptive Modulation and Coding
AQM	Active Queue Management
ARED	Adaptive Random Early Detection
BSD	Berkeley System Distribution
CDMA	Code Division Multiple Access
DNS	Domain Name System
ECN	Explicit Congestion Notification
FEC	Forward Error Correction
FIFO	First In, First Out
FTP	File Transfer Protocol
HARQ	Hybrid Automatic Repeat reQuest
HSDPA	High-Speed Downlink Packet Access
HS-DPCCH	High-Speed Dedicated Physical Control Channel
HS-DSCH	High-Speed Downlink Shared Channel
HS-PDSCH	High-Speed Physical Downlink Shared Channel
HS-SCCH	High-Speed Shared Control Channel
HSTCP	High-Speed TCP
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Taskforce
INRIA	Institut National de Recherche en Informatique et en Automatique
IP	Internet Protocol, originally Inter-network Protocol
ISO	International Organization for Standardization
ISP	Internet Service Provider
KKT	Karush-Kuhn-Tucker
KTH	Kungliga Tekniska Högskolan
OSI	Open Systems Interconnection

PI	Proportional Integral
RED	Random Early Detection
REM	Random Exponential Marking
RFC	Request For Comments
RNC	Radio Network Controller
RNF	Radio Network Feedback
RTO	Retransmission Timeout
RTT	Roundtrip Time
SACK	Selective Acknowledgement
SIR	Signal to Interference Ratio
STCP	Scalable TCP
TCP	Transmission Control Protocol
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications Service
WCDMA	Wide-band Code Division Multiple Access
WLAN	Wireless Local Area Network
XCP	eXplicit Control Protocol

Prologue

The Internet

Congestion control is one piece of the Internet machinery. But how does it fit in the larger picture, and how does the Internet really work? A younger second cousin of mine, Tom, once asked me how life was before the Internet. Let us meet again in a few years, and continue that conversation.

History

Tom: Could you use the Internet when you went to school?

Niels: No, I got my first account an a machine with a real Internet connection in October 1992, two years year after I entered university.

Tom: So that was the first time you used email?

Niels: Not quite, I got a student account at the cs department one and a half year earlier. The computer system for the students was not directly connected to the Internet, presumably for the protection of the Internet, but one could send and receive email, both locally, to other students and teachers, and to the outside world.

Since email was the first big application for the network, it was common with gateways between Internet email, and other systems. I remember having a summer job in the early 1990s at a telecommunications company. We had email, but no direct connectivity to the Internet. It was not possible to connect directly to ftp servers around the world, but some file archives offered an ftp by email service, which we used.

Tom: So how did people exchange files before the Internet?

Niels: Computer communication was fairly obscure; some people used modems to dial in to bulletin board systems, and hacker groups would exchange data on floppy disks sent via postal mail. It was common that computer magazines and books published program listings, but to try the program, you had to first type it into your computer by hand.

Tom: When did Sweden get connected to the Internet?

NIELS: I think the first network that was connected was the university network SUNET. A 56 kbit/s satellite link was set up between the Nordic university networks, and the John von Neumann Center at Princeton, New Jersey. Due to a delay on the American side of the link, Sweden missed the outbreak of the first Internet worm, released by Robert Morris on November 2, 1988[3]. The satellite link was operational a week later[4].

TOM: It's hard imagine how it was before the Internet, I use email and instant messaging constantly to keep in touch with friends.

NIELS: The phone system is of course many decades older than the Internet, so that's what people were using to keep in touch. Today, the Internet is used for everything, not just email and file transfer, but phone calls, radio, film distribution, etc. Before this convergence, we had a couple of large but separate communication systems, starting with radio broadcast and the (wired) telephone network, then television (different systems for terrestrial, satellite and cable), and finally mobile telephone networks.

Circuit switching vs packet switching

TOM: How does the Internet work, then? What makes the Internet different from the telephone network?

NIELS: The telephone systems were traditionally circuit switched. Originally, when making a phone call, you had an operator manually patching together a physical electrical circuit between the two telephones. Later on, this manual patching was replaced by automatic electromechanical switches, reacting to each digit of the phone number as you dialed, and then by digital systems.

TOM: This seems easy enough for local calls; there's a cord from each telephone to a huge switch board at the telephone switch, and to connect two telephones, you patch the corresponding two cords together. But how could you patch together a long distance call?

NIELS: For the duration of a call, you need a path reserved through the network, from one switch to the next. Between neighboring switches, your call might use a separate cord, a frequency band in a cable using frequency division multiplex, or certain time slots in digital system with time division. This means that every switch on the path must be aware of your call, and change their state when the call is set up or teared down.

TOM: I've heard that the Internet is "packet switched", what does that mean?

NIELS: Too see the difference between circuit switching and packet switching, say you want to transport fuel from the harbor at Värtahamnen to Arlanda airport.

[3]The worm infected VAX computers and Sun workstations. It is guessed to have infected 6 000 out of the roughly 60 000 computers connected to the Internet at the time.

[4]Kaarina Lehtisalo, *The History of* NORDU*net—Twenty-five years of networking cooperation in the Nordic countries*, `http://www.nordu.net/history/book.html`.

One alternative is to construct a pipeline between these two points. A different way to solve the problem is to load the fuel onto trucks, and let each truck find its way from the harbor to the airport independently. Pipelines are sure useful in certain circumstances, but using a general purpose road network is more flexible. You can freely mix vehicles of different sizes and with different cargo, while you can't use the same pipeline system to transport aviation fuel and beer.

Routing

TOM: But unlike trucks, packets have no drivers?

NIELS: There have actually been some research on "active networking", where each packet contains the intelligence needed for it to find its way through the network[5]. But in the Internet, packets are not smart enough to find their way by themselves.

TOM: So when I send a packet over the network, how does the network know where to send it?

NIELS: Each packet includes its source and destination address. These are the IP-addresses, 32 bits, usually written as, e.g., `130.236.254.103`[6]. Have you ever manually configured the IP address of your computer?

TOM: I think I had to to that sometimes, but that was many years ago, so I don't quite remember. Nowadays it's always automatic. But I remember having to type in some numbers for the address, and also for something called "netmask" and "default gateway", whatever that meant.

NIELS: That's the first step of routing. Your computer needs to know which addresses belong to computers on the same local network as you, and that's what the netmask is; it gives the size of the prefix identifying your local network. On the Lysator network, the machine where I usually read email has the address `130.236.254.103`, and the netmask is `255.255.255.0`. That tells the computer that all addresses starting with `130.236.254.` belong to computers on the same local network. This is the network prefix, and in this case, it's 24 bits long.

When I send a packet, the IP stack first uses the netmask to check if the destination address is on the same local network. If it is, the packet can be sent to the destination directly, without passing through any intermediate routers.

TOM: So that's what the netmask is for. And the "default gateway"?

NIELS: That's the IP address of a router, which must be located on the local network, that routes packets to and from the outside world. All packets sent to destinations beyond your local network, are sent to the default gateway, which then passes them on to other routers. It's called "default", because it is possible to set up more complex routing rules, and the default gateway is used when no other rule match.

[5]This goes back at least to the Softnet project in the early 1980s, developed by Jens Zander and Robert Forchheimer at Linköping University.

[6]For IP version 6, the addresses are 128 bits.

TOM: So how does the gateway know what to do with my packets?

NIELS: The routing system is more or less automatic. Each router has a couple of in- and outgoing links. It keeps track of which parts of the network are connected, directly or indirectly, to each link. Within an organization, routes can be configured manually, or via protocols such as OSPF (Open Shortest Path First), which exchanges routing information with neighboring routers. The OSPF protocol finds the shortest, or lowest cost, path between each pair of routers. Between organizations, routing depends not only on network topology, but also on private service agreements between Internet Service Providers (ISPs), basically, on who is paying whom. This routing information is exchanged between routers, taking policy into account, using BGP (Border Gateway Protocol).

No matter which protocol or method is used to configure the routing, the end result is a *routing table*, a large lists of network prefixes, and for each prefix, the outgoing link and neighbor router for that prefix. The destination address of each incoming packet is looked up in the routing table to find the longest matching prefix, and then the packet is transmitted to the corresponding router.

Routers close to the network edge often have a default gateway; a neighboring router which is more central, and which is used for all packets not matching any other rule. Routers in the core network don't have any default gateway, and this core is sometimes called the "default-free zone".

TCP/IP

TOM: When people talk about TCP/IP, what does that mean? Is it just a more complicated way to say "Internet"?

NIELS: Well, it's more or less the same thing. At least IP, which simply stands for the Internet Protocol.

If packet switching is analogous to a general purpose road network, then the IP protocol is the standardized freight container for intermodal transport. You pack some goods in a container. Your container is picked up by a truck, then loaded onto a train to a container terminal, then loaded onto a container ship, unloaded at some distant harbor, loaded onto another truck or train, and so on. All without any handling of the cargo inside the container. Before 1970, container transport suffered from several national or company-specific standards, all incompatible with each other. Intermodal freight transport took off after the ISO standardization of container sizes around 1970, transforming the world economy in unexpected ways[7].

TOM: So there were other packet switching protocols before IP?

NIELS: Sure. I'm not very familiar with any of them, but there were many proprietary networking protocols, e.g., SNA from IBM and DECnet from Digital Equipment

[7] Marc Levinson, *The Box—How the shipping container made the world smaller and the world economy bigger*.

Corporation. The x.25 protocol was standardized by CCITT (Comité Consultatif International Téléphonique et Télégraphique) in 1976, primarily for use in networks run by telephone companies. TCP and IP were developed during the 1970s, by the Defense Advanced Research Projects Agency (DARPA) in the United States, as communication protocols for interconnecting different networks. ISO standards for networking were developed during the 1980s, accepting x.25, but not TCP/IP, as a part of the Open Systems Interconnection (OSI) standard.

TOM: But TCP/IP is still not an ISO standard? So what happened with that?

NIELS: When NORDUnet were implementing wide area networking in the late 1980s, the only part of OSI networking standards that was in real use was x.25. And x.25 had both technical and economical problems[8]. So the choice was between the OSI standards, which were not mature, various proprietary networking technologies, and TCP/IP[9]. I imagine the considerations were similar at other organizations planning or building networks.

The first version of NORDUnet was a wide area Ethernet that supported all of IP, DECnet and x.25. Some year later, the network was upgraded to a pure IP network, but still with support for DECnet and x.25 on top of IP.

Architecture

TOM: So IP won the protocol war. Was that just due to good timing, or what is it that makes IP different?

NIELS: As the name implies, IP was designed for the interconnection of heterogeneous networks. I think it managed to find the right, close to minimal, interfaces for doing that. Transmission of IP packets is straightforward to implement on top of virtually any link technology[10]. And on top of IP, end points can implement more sophisticated communication and application protocols. The Transmission Control Protocol (TCP) provides a bidirectional reliable connection between two end points. Most application protocols you use are specified as working on top of any bidirectional data stream, and are used on top of TCP. Some that don't need a connection, or don't need reliable transmission, can work with IP directly[11].

TOM: Isn't that the obvious way to design a networking protocol?

[8] Quoting Hans Wallberg, manager of SUNET, "x.25 became terribly expensive when usage grew, since the cost was based on the amount of data transferred. The performance was also too poor. Even if you had a 64 kbit/s connection you never got more than 2400 bit/s due to all overhead."

[9] The so called "protocol wars" are described in some more detail in Kaarina Lehtisalo's book, which is one of my main sources for the history of the early years.

[10] An experimental specification for the transmission of IP packets with carrier pigeons, RFC 1149, was published on April Fool's day 1990. A decade later, this specification was implemented for the first time, in a cooperation between the Linux and BSD User Group in Bergen, and Vesta Brevdueforening.

[11] Actually, these protocols usually don't work with IP directly, but with the UDP protocol, which is a very thin layer on top of the best effort IP packet delivery service.

NIELS: It may seem obvious today. But for earlier networks, the applications and the link technologies were more tightly coupled. E.g., in the telephone system, all links and connections were of fixed bandwidth (around 4 kHz for analog transmission and 60 kbit for digital transmission), chosen for supporting a voice service of reasonable quality. And there were few applications besides voice.

TOM: But now your comparing computer network to the telephone system that is many decades older. That's cheating. What if you compare IP to DECnet or to the OSI protocols?

NIELS: I don't know many details about how DECnet works, but one important difference is that IP is an open, non-proprietary standard. You can buy devices that speak IP, network equipment, computers, various gadgets, from a large number of different companies, and they will work together. As for OSI, those protocols were complex committee products, where everybody's favorite feature was included. For example, the OSI model specifies two more layers than in TCP/IP, and the OSI network layer supports both datagram-oriented and connection-oriented services. In TCP/IP, the split between IP and TCP means that the network layer need not be aware of connections. The abstraction of a connection between two hosts is created by the TCP-implementation in the end points. This is an example of the end-to-end principle.

TOM: The "end-to-end principle"? What in the world is that?

NIELS: The end-to-end principle states that in a communications system, as much as possible of the protocol logic and state should be located in the communication end points. This is important for the scalability of the system, and one of the most important principles for the architecture of the Internet. In practice, one direct consequence is that a TCP connection can survive a reboot of routers along the path, as well as routing changes.

TOM: Quite different from the old telephone system.

NIELS: In the telephone system, the telephone devices are simple, and all the complexity is in the network. In the Internet, the network layer is stupid, and instead the end points have to be quite complex[12].

Congestion control

TOM: When driving, I often get stuck in the traffic, waiting and waiting in some queue. Do packets get stuck too, on the Internet?

NIELS: That's congestion; when there is more traffic than the network can handle. In the Internet, each router has some incoming and some outgoing links of limited capacity. For example, if a router has two incoming links where 80 Mbit of packets

[12] But not as complex as one might think. Adam Dunkel's TCP stack `uip` supports 8-bit micro controllers, and needs about 5 Kbyte for code and a few hundred bytes of RAM, depending on the application.

arrive every second, and all packets are routed to the same outgoing link, with a capacity of only 100 Mbit/s, then that router is overloaded. Routers buffer packets at the outgoing link, so when the router is overloaded, packets are queued up in that buffer. But unlike traffic queues, the buffer size, and hence the queue size, is limited. When the limit is reached, arriving packets are not added to the queue, they are thrown away or "dropped onto the floor".

TOM: Not quite like road traffic then.

NIELS: No, it's as if there were trapdoors in the roads, located some 100 m before each intersection. The trapdoor leads to a bottomless hole, and opens automatically whenever there's a queue all the way from the trapdoor to the intersection.

Another important difference compared to the road network is the traffic pattern. The bulk of the traffic on the Internet is transfer of fairly large files. A typical file will be divided into somewhere from a hundred packets for a moderate size image, to several thousands of packets for larger files. All these packets have the same source and destination addresses.

TOM: So when my computer sends a file, it transmits a thousand packets straight away?

NIELS: With the earliest versions of TCP, you might almost have done that. The TCP protocol included flow control from the start; this mechanism lets the receiver tell the sender how much data the receiver can handle, and the sender must not send more than that. In the road traffic analogy, you tell the sender that you have free parking space for only ten trucks, and then the sender will load at most ten trucks and send them your way. When the trucks have arrived, been unloaded, and left again, you tell the sender that you have new parking space available.

TOM: So if the receiver has space for a thousand packets, I send that?

NIELS: That could cause problems for the network. As soon as the capacity of the computers connected to the network outgrow the capacity of the routers, the network suffered "congestion collapse", with drastically reduced performance. This was during the Internet's infancy in the early 1980s[13]. It became clear that flow control was not sufficient. It prevents overloading of the receiver, but not overloading of intermediate routers. Congestion control was needed too.

TOM: So how does that work?

NIELS: The sender maintains a limit, called the *congestion window*, for the number of packets in-flight in the network, i.e., packets that have been sent but not yet acknowledged. Say the congestion window is five packets. Then you will have five packets in-flight somewhere in the network, waiting for an acknowledgement (ACK) saying that a packet has arrived to the destination and exited the network. And you send a new packet only in response to an ACK for some older packet.

Then TCP also has rules for the adjustment of this window size: When the connection is new, the window size is increased by one packet for each received

[13]See RFC 896 for a contemporary source

ACK, meaning that you can send two new packets for each received ACK. When a packet is lost, meaning that the network is getting congested, the window size is cut in half, followed by an increase of one packet per roundtrip time. This style of congestion control was introduced in TCP around 1988, and it seems to work fairly well.

TOM: That sounds like really simple rules. Amazing that it works so well. Anyway, there's one other thing I don't understand. You have been talking a lot about IP and IP-addresses. But you never see any IP-addresses, do you? At least all addresses I use are human-readable, e.g., `wikipedia.org`. Are they related in some way?

NIELS: That's the Domain Name System (DNS), which is a huge distributed database. But it's getting late, so if you'd like me to try to explain how that works, let's do that over dinner.

Chapter 1

Introduction

WINDOW-BASED congestion control is an Internet work horse. Up to 90% of the traffic is managed by the TCP protocol, used for web browsing, file-sharing, email transmission, and innumerable other applications. The remaining traffic serves applications such as voice over IP, online gaming, and Domain Name System (DNS) service. To a first approximation, Internet traffic can be divided into two types: TCP traffic and real-time traffic.

The TCP traffic uses window-based congestion control, which adapts each flow's average sending rate to the flow's fair share of available resources. The collection of TCP flows tries to use all available capacity, but it also responds to network congestion, and will reduce each flow's sending rate if the network load increases, or if the capacity in the network is reduced.

The real time traffic uses different mechanisms for congestion control, or none at all, together with explicit or implicit admission control[1]. Which means that the sending rate of a real-time application is mostly independent of the current network state. The reason that these real-time application has, so far, not caused any Internet breakdown, is in part due to the admission control, and in part due to the fact that the typical real-time application does not need high capacity[2]. In the current Internet, the quality of real-time applications degrade severely if they share a bottleneck with TCP flows.

This chapter is organized as follows. First, in Sec. 1.1, we look into the future, to see what the Internet will be like, and what will be needed to get there. Sec. 1.2 gives a brief introduction to IP networking and the window-based congestion control of TCP. In Sec. 1.3, we describe the control objectives for Internet congestion control. An overview of the contributions in this thesis is given in Sec 1.4. Publications are listed in Sec. 1.5, and finally, Sec. 1.6 provides the outline for the rest of the thesis.

[1] For an example of implicit admission control, consider an online game. If delay or packet loss to a game server becomes too bad due to congestion, the player will likely quit the game and try at a later time.

[2] A voice call needs at most 60 kbit/s, so a 100 Mbit/s Ethernet, a common capacity in home and office networks, can easily accommodate several hundred simultaneous voice calls.

Figure 1.1: The improvised headquarters for the emergency response team is located in the high "Lipstick" building in Gothenburg, at the shore of Göta Älv. Photo, *Painting the sky with lipstick*, by Jonas Ahrentorp.

1.1 Motivation

An emergency response scenario

Gothenburg, November 2015. A week ago, the embankments around the harbor and the city center broke down; they could not hold against the severe autumn storm, and the higher sea level we have seen in recent years. Since the sea water flooded several underground tunnels, transformer stations, telephone switches, and Internet peering centrals, the basic infrastructure is still not working (not to mention the flooded sewer system).

The city center, roughly from the harbor up to Vasagatan, is flooded. Still, the area is far from deserted, a few hundred people are working there, local emergency response personnel, additional professionals from Uddevalla and Copenhagen, together with volunteers who live or work in the area and decided to stay and help.

The center of all this activity is the well-known "Lipstick" building (Fig 1.1). When the electricity and the telephone system stopped working, the people working in one of the offices half-way up in the tower realized that the location was ideal for wireless communication to large parts of the flooded area. Enabling ad-hoc networking, they soon got connected to other people living or working nearby, who were also cut off from the outside world. The network grew to about 150 active nodes, and soon somebody managed to get in contact with emergency response

people.

Within the hour, a military helicopter landed on the roof[3]. The passengers were a couple of emergency response workers from Mölndal's fire-station, who brought with them a microwave link and a small diesel-powered generator, and sufficient fuel for a couple of days. As soon as the microwave was operational, the ad-hoc network had a reasonable connection to the Internet.

The network was used primarily for voice calls and email, first locally, and, once there was connectivity to the outside, to all of the world. People could get in contact with relatives, and the emergency response people could get in contact with the people in the area to assess the situation and organize reinforcements.

The next few days, things went smoothly. Evacuation, by boat or helicopter when necessary, was organized. Patients waiting for evacuation were equipped with networked pulse oximeters and other sensors, monitored both by local medical staff and the medical support center at the Sahlgrenska University Hospital. The coordination office moved down to the bottom of the tower, since it proved difficult to use the temporary generator to power the tower's elevators. The people from Uddevalla and Copenhagen arrived, and they started to work, almost not noticing how their computers automatically joined the communication network. The support for those who stayed was organized, based around the Lipstick tower: Meals, temporary hygiene facilities, as well as electrical charging stations for everybody's computers. Food and fuel were delivered by boat from Kungälv.

The demands on the network changed gradually during this week. The whole time, voice and email were the most important applications, but there were also sensor data, transfer of photos and videos documenting the situation, and file-sharing of all sorts of useful information, including large files with detailed blueprints for buildings and infrastructure.

This is the first time in Sweden that Internet communication has played such an important rôle in an emergency situation, but it is not unique in the world. Two years ago, in...

Issues for the evolution of the Internet

There are several key issues that need to be addressed, to make such a network a working reality.

Ubiquitous wireless networking All devices in this story use wireless networking. All connectivity is subject to large variations in quality and capacity.

Mobile ad-hoc networking To get connectivity between each pair of nodes, every node may need to forward packets on behalf of other nodes. The topology is ever-changing, as the users and their computers move around.

[3]The helicopter platform was added to the building a few years earlier, when the headquarter of the Swedish Sea Rescue Society moved from Långedrag to new offices in the tower.

Zero configuration In this story, most of the people are not computer experts. And those few who are, do not have time to configure the network settings of all the others. Furthermore, the people involved include people from various Swedish and Danish government organizations and private citizens. This means that there's no single central IT support department that has configured the involved computers in advance. Things must just work.

Mix of real-time traffic and large file transfers The most important application on the network is voice calls, which have intrinsic real-time requirements. At the same time, the network is shared with other applications, in particular peer-to-peer file-sharing, which keep the network load close to capacity. The combination of high load, and capacity variations can easily lead to congestion, with packet losses and large queueing delays, which may cause unacceptable quality degradation for the real-time voice calls.

Time varying wireless channels, the changing topology of the ad-hoc network, and the requirement of minimal configuration, rule out complex quality-of-service mechanisms. There is no place for service level agreements, in-advance reservation of resources, complex policy configuration, etc. Simplicity is essential.

All of these issues, except possibly the need for ad-hoc networking, are important issues also for the everyday Internet in coming years. The central theme in this thesis concerns the last of these issues. We model the the queueing dynamics in a network with both traffic subject to window-based congestion control, and real-time traffic, and analyze its dynamical properties. We can then use model-based design to construct congestion control protocols that maintain small queueing delay.

By reducing the queueing delay, we can provide high quality for both real-time applications and peer-to-peer file transfers, and at the same time have full utilization of available capacity.

1.2 Overview of networking

Network layers

A communication network is a set of links and nodes. The set of nodes can be divided further into routers, which forward packets between links, and hosts, which are the communication end points. On the physical layer, common link types are based on optical fibers, copper cables, or radio transmission. When more than one node can transmit on the same link, the use of the link is coordinated using some medium access control. This coordination, together with the conventions for how to encode an IP-packet for a particular link type, is referred as the *link layer*. The service provided by the link layer is the transmission of IP packets between nodes that are connected to the same link.

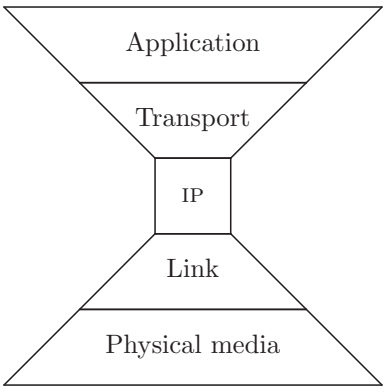

Figure 1.2: The layers of the IP networking model. The IP network is implemented on top communications links based on many different link technologies. Above IP, there are a couple of widely used transport protocols, and an innumerable number of different applications.

To provide global connectivity, forwarding of packets between links is necessary. Deciding which path each packet should take between its source and destination is the responsibility of the routing system. Routing, together with the addressing architecture, are the main services of the *network layer*. The network layer provides best effort delivery of IP packets to and from arbitrary nodes in the network.

IP is the underlying infrastructure, shared by all Internet applications. However, many applications need a more sophisticated communication service than best effort packet delivery. This is built on top of IP, in the form of transport protocols. Services provided by transport protocols include

- Addressing of processes running on the same host, sharing the same IP address.
- Acknowledgements of received packets, and automatic retransmission of lost packets.
- The notion of a *connection* between two processes running on two nodes in the network.

Widely used transport protocols are the User Datagram Protocol (UDP, [97]), and the Transmission Control Protocol (TCP, [39]). The UDP protocol is basically an IP packet delivery service, but with an extra port number attached to the addresses of source and destination. The port number identifies a process or application running on the host.

The most important transport protocol is TCP, which provides for a bidirectional stream of data between two nodes. The data stream is divided into segments that are transmitted as IP packets. Sequence numbers and acknowledgements are used, to

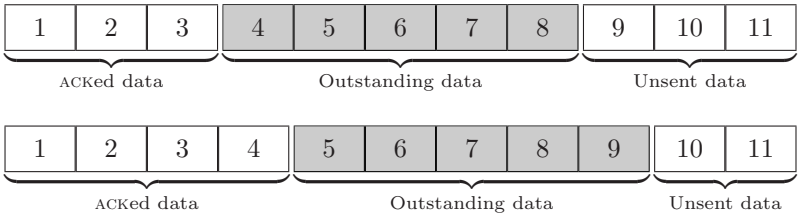

Figure 1.3: Sliding window control. The window size is the amount of outstanding data. In this example, it is five packets. In the top figure, eight packets have been transmitted, and ACKs for the first three packets have been received. The bottom figure illustrates the situation after the ACK for the fourth packet has been received. One more packet, the ninth, is transmitted, and the window of outstanding data "slides" forward one packet.

reassemble the stream at the other end, and to detect if packets are lost, reordered or duplicated by the network. The protocol also implements *flow control*, which prevents the sending node from overloading the receiver, and *congestion control*, which prevents the sending node from overloading the network. In a recent study of the traffic mix in Deutsche Telekom's ADSL network [54], more than 90% of the traffic was using TCP, and a significant proportion thereof was peer-to-peer file-sharing.

Everything on top of the transport layer is referred to as *application layer*. There are innumerable application protocols, from the older `telnet` protocol and Simple Mail transfer Protocol (SMTP), to the Hypertext Transfer Protocol (HTTP) used for the World Wide Web and the Session Initiation Protocol (SIP) used to set up IP telephony calls. Some general purpose application protocols, such as Transport Layer Security (TLS, also known as SSL), and the transport part of the Secure Shell protocol suite, can be thought of as new transport protocols, even though they are built on top of TCP.

The layering architecture is illustrated in Fig 1.2. This layering, with IP as the common language, implies that any application can work over any heterogeneous networking path[4]. Usually, an application need not be aware of what type of links it is using, and a link need not be aware of the type of applications that are using it. This fundamental principle of separation can be compared to Shannon's separation of source and channel coding, and the von Neumann computer architecture[64].

Window-based congestion control

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

[4]The model shown here is a simplified version of the OSI reference model, with its seven layers.

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. This is illustrated in Fig. 1.3.

Window-based congestion control was originally motivated by the fluid flow analogy.

> *... 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, [59]

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 (RTT) fluctuates, the sliding window gives an average sending rate of one full window per average RTT. The window size is adjusted depending on received ACKs, and it is the details of this outer-loop that differ between TCP variants.

The most widely used version of TCP is New Reno, which uses the following window update rules. As long as there are no packet losses, the window size is increased by one packet per RTT. And when a packet is lost, the window size is reduced to one half of its previous value. The linear increase of the window size, in the absence of packet losses, together with the multiplication by 1/2 when a packet loss is detected, is called Additive Increase, Multiplicative Decrease (AIMD).

This thesis proposes a new model for the queueing dynamics under window-based congestion control. The dynamic behavior depends on the amount of real-time, non-congestion controlled, cross-traffic that the network is shared with. The main novelty in the model is to accurately describe the influence of this cross-traffic, and to note that this influence is significant, in terms of central system properties such as gain and time constants.

1.3 Problem statement

The overall goal of congestion control is to optimize the performance in a communication network. This optimization means, roughly, that sending rates at the data sources should be as high as possible, without overloading the network. The primary measure of network overload is packet losses; when the arrival rate at a link exceeds capacity, the corresponding queue starts to build up, and when the queue is full, packets must be discarded. The bottleneck links in the network should be fully utilized. The requirement of a small loss rate implies that the average arrival rate at each bottleneck link should either match the link capacity exactly, or be very slightly larger.

When the network is shared with real-time traffic, it becomes important not only to maintain a small packet loss rate, but also to maintain reasonably small queues, since large queues imply large delays.

By the term *congestion control*, we include both the rules for adjusting window sizes and sending rates which are implemented in end nodes, and all related rules and mechanisms, such as Active Queue Management schemes (AQM) and Explicit Congestion Notification (ECN), which are implemented in routers.

Control objectives

Congestion control has been a very active area of research during the last decade. Still, the current transport control protocol TCP works fairly well. So why change that which is not broken? To understand that, we must first understand what the control objectives are:

Avoid network overload: The probability of packet loss due to congestion should be small.

Efficient resource utilization: All bottleneck links should be fully utilized.

Fair sharing: The congestion control mechanism determines how resources are shared between users. The sharing should be predictable and satisfy some reasonable notion of fairness.

React to changes: The network load is constantly varying, there are occasional routing changes, and some (wireless) links have varying capacity and delay. The congestion control must react to these changes and adjust the sending rates.

Small queueing delays: Both large queues and large queue fluctuations can harm other applications using the network, in particular real-time applications.

Of these objectives, TCP achieves all but the last one, when used over the ordinary wired network for which it was designed. For low-power wireless links, and for links with large propagation delay or large bandwidth-delay product, TCP also has difficulties achieving efficient resource utilization. We therefore see three main problems that are being addressed by current TCP research:

TCP over wireless: TCP interprets packet losses as a sign of congestion. If a wireless link loses some packets due to disturbances on the radio channel, TCP reduces its sending rate even when there is no real congestion. Many wireless links tries to avoid this problem by retransmitting lost packets at the link layer. However, this leads to larger delays and larger delay variations, which are also problematic for TCP.

High capacity, large delay networks: To use available capacity efficiently, TCP needs a window size close to or larger than the bandwidth-delay product. In the basic TCP protocol, the window size is limited to 64 Kbyte, which is quite small with todays high speed links. A 100 Mbit/s link with 5 ms RTT has a

bandwidth-delay product of 61 Kbyte. To overcome this problem, one can use the window scaling option [60].

But even when window scaling is used, so that the window size can grow to match the bandwidth-delay product, the AIMD rules imply that it takes a large number of RTTs for the window size to grow large enough. If also the RTT is large, this leads to poor throughput.

Reducing queueing delay: Router buffers are usually quite large, on the order of 100 ms times the link capacity. The additive increase rule implies that queues at bottleneck links will grow to fill the available buffer. At this point, the router must drop some packets. When end hosts detect these losses, they reduce their sending rates, and the queue can start to shrink. This implies that the queueing delay at a bottleneck will oscillate between a maximum value on the order of 250 ms, and a minimum value that, among other factors, depend on the number of flows. These delays are problematic for any real-time application sharing the same link.

Deployment constraints

Furthermore, for a congestion control algorithm to be deployable over the Internet, we have additional constraints:

Scalability: The algorithm must work for huge networks, in terms of the number of nodes, links and data flows. The amount of resources, e.g., memory and computation power, required in any single node in the network, must not grow with the size of the network topology, or with the number of concurrent data flows.

End-to-end principle: As much as possible should be done at the communication end points. In particular, we cannot require network core to keep any per-flow state.

Robustness: Parameters such as number of flows, capacities, and delays vary over ranges of several orders of magnitude, and knowledge of the parameters is very limited. Hence, the algorithm must work despite severe uncertainty about the parameters.

Tunability: The number of tuning parameters should be small, and the effect of each parameter should be predictable and easy to understand. For scalability, the parameters in each node should be tunable based on *local* information. E.g., to tune the parameters of an AQM scheme for a particular link, it should be sufficient to observe local properties such as the queue size and arrival rate at that link.

Incremental deployability: A new algorithm must work in the setting that a subset of end-nodes and routers are upgraded, and flows using the old and

the new algorithm share resources. It is not feasible to have a flag day when all end nodes or all routers switch to a new algorithm.

Openness: One of the reasons for the success of the Internet was that the protocols were free to implement for anyone. Most people in the IETF (Internet Engineering Task Force) and W3C (World Wide Web Consortium) communities understand that patent encumbered protocols are not suitable for use in Internet standards. Patenting a network protocol almost surely guarantees that the protocol will never see any wide use on the Internet[5].

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 RTTs 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 [101].

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.

1.4 Main thesis contributions

The contributions of this thesis address the modeling, analysis, and design, for improved congestion control over the Internet.

[5]For a sad example, see all the great research in secure password authentication. That is an urgent problem which is technically solved, yet nobody is using the solutions because the entire area is a patent swamp.

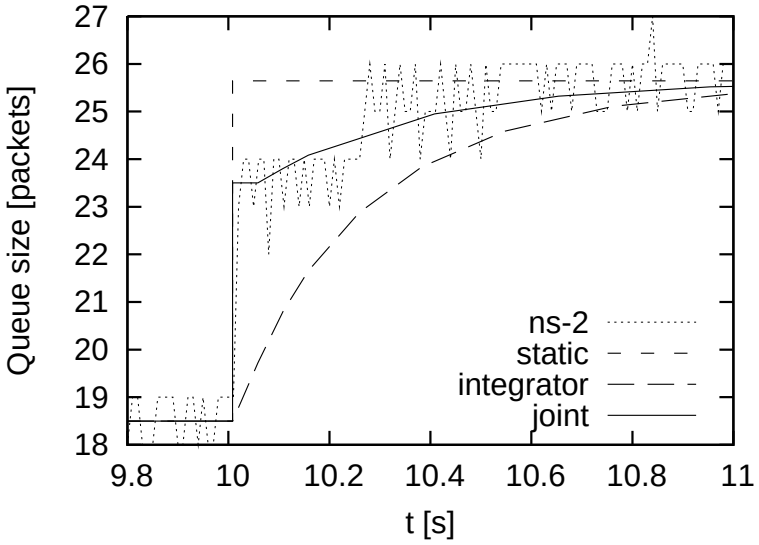

Figure 1.4: Queue response from `ns2` compared to the prediction of three queue models. The simulation setup consists of a single flow with window-based transmission control, sharing a single bottleneck with 30% inelastic cross-traffic. At time $t = 10$, the window size is increased by five packets.

Modeling

In Chapter 3 we develop and validate an improved model for the feedback system consisting of a queueing network, and data sources using window-based congestion control. The window sizes are the system inputs, and the queue sizes are the system outputs. This model describes the inner-loop of all TCP-like window-based congestion control methods. The main novelty of the model is that it takes into account that the network is shared with traffic which is not subject to congestion control, such as video streaming and voice over IP. The amount of such cross-traffic has a large influence on the dynamics of the system, in particular the gain and the time constant.

A step response is shown in Fig 1.4. The dashed curves represent two models, the static model [114] and the integrator model [56, 80], that have been used in the literature. The dotted curve is the result from `ns2` simulations, and the solid curve is the new model. The amount of cross-traffic is an important parameter for the dynamics; in this figure, there is 30% cross-traffic.

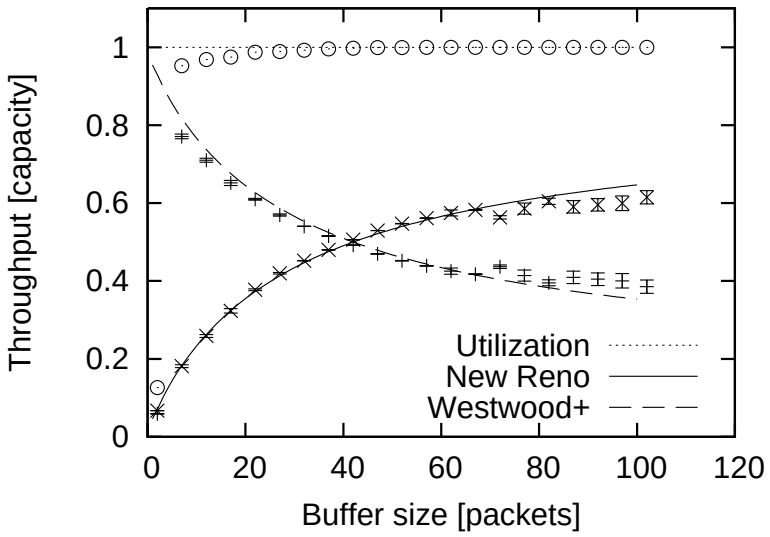

Figure 1.5: Normalized throughput for TCP New Reno (increasing curve) and Westwood+ (decreasing curve), as a function of the buffer size at the bottleneck router. The topmost almost flat curve is the link utilization.

Analysis

Chapter 4 analyzes the stability properties of the inner-loop in window-based congestion control, as well as the convergence time.

Chapter 5 analyzes the fairness when flows using two different flavors of TCP, New Reno and Westwood+, compete for bottleneck capacity. This will also give us an opportunity to look closer at how AIMD works, and how it relates to buffer sizing issues. As seen in Fig 1.5, the sharing depends on the buffer size at the bottleneck. The two flows share the link equally only when the buffer size equals the bandwidth-delay product, which in this case is 42 packets.

Design

In Chapter 6, we design a new congestion control mechanism, as an outer-loop around the inner-loop that was modelled earlier. The aim is to maintain the efficiency and fairness properties of TCP, but with significantly smaller bottleneck queues. The key ideas are to take advantage of the stability of the inner-loop, and to use rules for setting and reacting to packet marks that results in more frequent feedback than with AQM and ECN. Figure 1.6 shows the response in window size and queue size to a change in the cross-traffic intensity, for TCP New Reno and for the new mechanism.

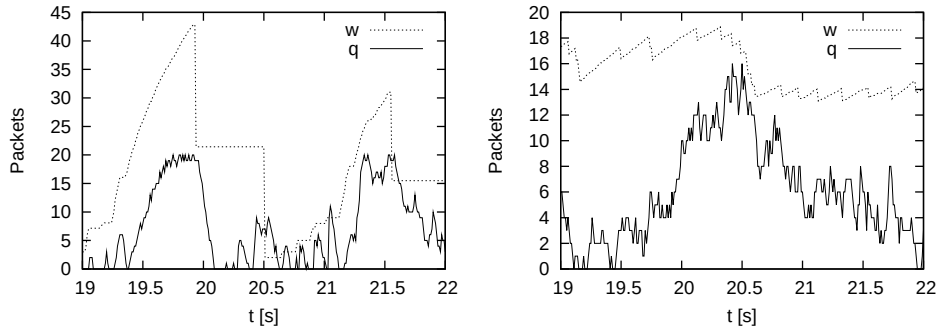

Figure 1.6: Window size (w) and bottleneck queue size (q) for a short segment of a simulation with a single bottleneck and a congestion controlled flow sharing the bottleneck with Poisson cross-traffic. At $t = 20$ s, the cross-traffic intensity is increased from 20% to 40% of link capacity. To the left: New Reno. To the right: The proposed mechanism. Note the different scale on the vertical axis.

Chapter 7 considers file download over the cellular network, and proposes a control mechanism that takes advantage of cross-layer signalling from the Radio Network Controller (RNC) in the radio network. The new controller improves user response time, queueing delay, and the utilization of available radio resources.

1.5 Publications

Congestion control

Modelling and estimation of network roundtrip times, for wired and wireless networks, is discussed in:

> Krister Jacobsson, Niels Möller, Karl Henrik Johansson, and Håkan Hjalmarsson. Some modeling and estimation issues in control of heterogeneous networks. In *Mathematical Theory of Networks and Systems*, Leuven, 2004.

I was fortunate to spend a couple of months during spring 2006 as a guest at INRIA in Sophia Antipolis, working with analysis of TCP Westwood. The results of this cooperation were published as:

> Eitan Altman, Chadi Barakat, S. Mascolo, Niels Möller and J. Sun. Analysis of TCP Westwood+ in high speed networks. In *International Workshop on Protocols for Fast Long-Distance Networks*, Nara, 2006.
>
> Niels Möller, Chadi Barakat, Konstantin Avrachenkov, and Eitan Altman. Inter-protocol fairness between TCP New Reno and TCP West-

wood+. In *Conference on Next Generation Internet Networks*, Trondheim, 2007.

The results from the latter paper are included in Chapter 5.

Back at KTH, I and Krister Jacobsson developed the joint link model for window-based congestion control, to better capture the dynamic properties of the ACK-clock. The model was first presented in:

Krister Jacobsson, Håkan Hjalmarsson and Niels Möller. ACK-clock dynamics in network congestion control—an inner feedback loop with implications on inelastic flow impact. In *IEEE Conference on Decision and Control*, San Diego, 2006.

With this improved model for the inner-loop of window-based congestion control, it was a natural next step to investigate the design of the outer-loop. The result is the congestion control protocol described in Chapter 6, which is being written up for journal publication as:

Niels Möller and Karl Henrik Johansson. Congestion control for small queueing delay. Manuscript in preparation.

TCP over wireless

Modeling of link-layer feedback systems such as power control and link-layer retransmissions is studied in:

Niels Möller and Karl Henrik 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.

Carlo Fischione, F. Graziosi, F. Santucci, Niels Möller, Karl Henrik Johansson, and Håkan Hjalmarsson. Analysis of TCP over WCDMA wireless systems under power control, MAI and link level error recovery. In *IWCT*, 2005.

Niels Möller, Carlo Fischione, Karl Henrik Johansson, F. Santucci, and F. Graziosi. Modeling and control of IP transport in cellular radio links. In *IFAC World Congress*, Prague, 2005.

Based on these models, the following paper proposes adding carefully chosen artificial delays to certain packets, to reduce spurious timeout in TCP, and improve end-to-end performance:

Niels Möller, Karl Henrik Johansson, and Håkan Hjalmarsson. Making retransmission delays in wireless links friendlier to TCP. In *IEEE CDC*, Bahamas, 2004.

These results on wireless links were also presented in my licentiate thesis, and are not included in the current thesis.

The cross-layer design described in Chapter 7 is a result of cooperation with Ericsson, producing a series of papers:

> Inés Cabrera Molero, Niels Möller, Justus Petersson, Robert Skog, Åke Arvidsson, Oscar Flärdh, and Karl Henrik Johansson. Cross-layer adaptation for TCP-based applications in WCDMA systems. In *IST Mobile & Wireless Communications Summit*, Dresden, 2005.
>
> Niels Möller, Inés Cabrera Molero, Karl Henrik Johansson, Justus Petersson, Robert Skog, and Åke Arvidsson. Using radio network feedback to improve TCP performance over cellular networks. In *IEEE Conference on Decision and Control and European Control Conference*, Seville, 2005.
>
> Niels Möller, Åke Arvidsson, Justus Petersson, Carlo Fischione, Robert Skog, Patrik Karlsson and Karl Henrik Johansson. Supporting end-to-end applications over HSDPA by cross-layer signalling. In *IEEE Wireless Communications & Networking Conference*, Hong Kong, 2007.
>
> Marco Fiorenzi, Daniele Girella, Niels Möller, Åke Arvidsson, Robert Skog, Justus Petersson, Patrik Karlsson, Carlo Fischione and Karl Henrik Johansson. Enhancing TCP over HSDPA by cross-layer signalling. In *IEEE Globecom*, Washington, 2007.

My interest in stability of window-based congestion control was initially motivated by this project. The stability results in Chapter 4 were presented in:

> Niels Möller, Karl Henrik Johansson and Krister Jacobsson. Stability of window-based queue control, with application to mobile terminal download. In *International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, 2006.

The simulation studies presented in Chapter 7 are the work of a large number of people, including three master of science projects under my supervision,

> Inés Cabrera Molero, Radio network feedback to improve TCP utilization over wireless links. Master Thesis, KTH, 2005.
>
> Marco Fiorenzi, Downlink TCP proxy solutions for interactive gaming over HSDPA and 3G Networks. Master Thesis, KTH, 2007.
>
> Daniele Girella, Downlink TCP proxy solutions over HSDPA with flow aggregation and user behavior. Master Thesis, KTH, 2007.

A journal article on the results of the radio network feedback project is under preparation.

> Niels Möller, Åke Arvidsson, Justus Petersson, Carlo Fischione, Robert Skog, Patrik Karlsson and Karl Henrik Johansson. Radio Network Feedback: Architecture and performance. Manuscript in preparation.

Other research

I have also had the opportunity to study some problems in the area where computer science meets computational number theory, which is unrelated to the topic of this thesis. The following contributions consider the problems of efficient computation of the greatest common divisor of two very large (10 000 digits or more) integers:

> Niels Möller. On Schönhage's algorithm and subquadratic integer gcd computation, In *Mathematics of Computation* (77) 589–607, 2008.
>
> Niels Möller. Robust HGCD with no backup steps. In *International Congress on Mathematical Software*, Castro Urdiales, 2006.

1.6 Thesis outline

Chapter 2 describes the principles of TCP/IP and the mathematical tools needed for understanding Internet congestion control. We also give an overview of current research in congestion control, and of the issues when using TCP/IP over wireless links.

In Chapter 3, we develop and validate a model for the feedback system consisting of a queueing network, and data sources using window-based congestion control. The window sizes are the system inputs, and the queue sizes are the system outputs, so this system is the inner-loop of all TCP-like window-based congestion control methods. The main novelty of the model is that it takes into account that the network is shared with traffic which is not subject to congestion control, such as video streaming and voice over IP. Chapter 4 uses this model to analyze the stability properties, as well as the convergence time.

In Chapter 5, we examine the fairness issue, when a bottleneck link is shared between flows that use different versions of TCP. This will also give us an opportunity to look closer at Additive Increase, Multiplicative Decrease, and how it interacts with buffer sizing issues.

In Chapter 6, a new congestion control mechanism is designed, as an outer-loop around the inner-loop that was modelled earlier. The aim is to maintain the efficiency and fairness properties of TCP, but with significantly smaller bottleneck queues. We establish stability of the control mechanism, in the fluid-flow setting, and investigate its behavior in packet-level `ns2` simulations.

Chapter 7 considers the TCP-over-wireless problem, in particular, file download over the cellular system, where the link capacity is time-varying. The proposed solution is a modified outer-loop, which takes advantage of cross-layer signalling from the radio network.

Finally, Chapter 8 discusses the results, and summarizes what we have learnt about modeling, analysis and design of congestion control protocols.

Chapter 2

Background

The Internet is competing with the global telecommunications infrastructure for the title "The biggest machine in the world"[1]. Understanding how it works may seem like a daunting task. In this chapter, we start with the principles of TCP/IP, the work horse protocols of the Internet. Sec. 2.1 describes the layered Internet architecture, and Sec. 2.2 explains how the TCP protocol works. The Prologue gives a broader and less formal introduction to the Internet's architecture and principles.

After this background, Sec. 2.3 describes the mathematical tools and frameworks that are used to analyze the system; from detailed models that are used for small networks, to frameworks that provide an understanding of the structure of large complex networks. Sec. 2.4 surveys the most important lines of TCP research, with a focus on window-based protocols.

Of the control objectives, the requirements on fairness and small queueing delays are related to the system's equilibrium. In Sec 2.5, we discuss the notion of fairness of congestion control protocols. In Sec. 2.6, we discuss queueing delay, and the related area of quality of service.

Finally, in Sec. 2.7, we describe the characteristics of wireless links, and the issues that arise when using TCP over wireless.

2.1 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 directly above and below at the *local* node, but only to the *same* layer at remote nodes. For example, 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.

[1] This phrase was used on Ericsson posters in the 1980's.

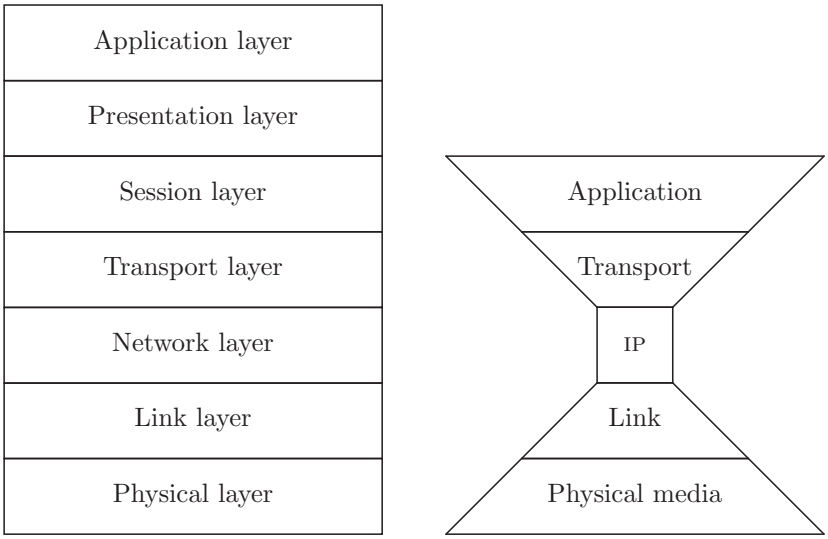

Figure 2.1: Left: The OSI networking stack. Right: The IP networking model.

The Open Systems Interconnection (OSI) reference model specifies seven layers, illustrated to the left in Figure 2.1. 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 technologies, such as 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 2.1.

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 the 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*, and it differs from many earlier network architectures [105, 49]. In particular, it contrasts to the architecture of traditional wired telephony system, where the telephone device at an end point can be built of just a dozen analog components.

2.2 Transmission control protocol

As described in Chapter1, two of the main objectives 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. The TCP protocol was developed in the late 1970s, resulting in the Internet Standard RFC 793 [39]. The principles for TCP congestion control were developed a few years later, in response to experience of "congestion collapses" in the Internet [92, 59].

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 window. The standard way of doing this is called New Reno [5, 45]. It is described in this section. One common extension is TCP with selective acknowledgements (SACK) [88, 18]. Before explaining the control mechanisms, we have to look into how TCP detects packet loss.

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.

When duplicate ACKs are observed on the sender side, there are several possible causes: a packet may have been delayed and delivered out-of-order, a packet may have been lost, or an ACK packet may have been 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 when a timeout occurs, by the exponential back-off mechanism.

TCP congestion control state

There are four distinctive states in the TCP congestion control, illustrated in Figure 2.2, and two state variables related to congestion control: The congestion window `cwnd` and the slow start threshold `ssthresh`. The value of `ssthresh` determines the window increase rule; if `cwnd` $<$ `ssthresh`, TCP increases `cwnd` by one packet for each received ACK (*slow start* state), and if `cwnd` $\geq$ `ssthresh`, TCP increases the window size by one packet per RTT (*congestion avoidance* state). Typical initial values when TCP leaves the idle state and enters the slow start state are a `cwnd` of 2 packets, and an infinite `ssthresh`.

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

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

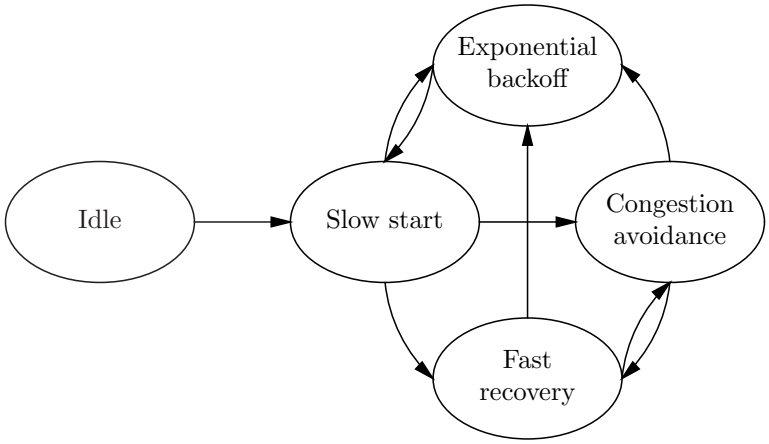

Figure 2.2: TCP state diagram. Transitions back to the idle state are omitted.

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 back-off 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 the *congestion avoidance* state, `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 back-off 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 back-off

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

- The lost packet is retransmitted.
- The state variables are updated by `ssthresh` ← `cwnd`/2, `cwnd` ← 1 packet.
- The RTO value is doubled.

When an ACK for the retransmitted packet is received, TCP enters the slow start phase. The above update of `ssthresh` implies that, in the absence of further packet losses, TCP will switch from the slow start phase to the congestion avoidance phase once `cwnd` is increased to half its value before the timeout.

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 [40]. The upper bound for the RTO is on the order of one or a few minutes.

Exponential back-off 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 back-off 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` ← `cwnd`/2. The reason for the `ssthresh` update is arrange so that the later window increase from `cwnd`/2 and up will use the additive increase of congestion avoidance, not slow start.

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 [5], 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 back-off 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, the original fast recovery procedure of TCP Reno is limited in that it can recover only one packet per RTT. This is the main problem addressed by both TCP New Reno [45] and TCP SACK [88, 18].

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 actual number of outstanding packets match the new window size.

The ACK clock

When TCP's congestion window is kept constant, the sender transmits one new packet for each received ACK. This 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 average transmission rate is one window of data per RTT. In TCP congestion control, the control signal is the window size, and the actual sending rate is controlled only indirectly by adjusting the window. This can be seen as a cascaded control system, where the ACK-clock is a fast inner-loop, operating on a per-packet timescale, which is combined with an outer-loop that adjusts the window size, and operates on an per-RTT timescale.

2.3 Mathematical models for congestion control

One crucial issue in model-based control design is to select a model with the right level of detail. The model should capture the important dynamics of the real system, and at the same time be simple enough for mathematical analysis of the system. The models for congestion control, and related tools, can be classified as:

Packet-level: A packet-level model accounts for the location of each individual packet, as the packets are queued and forwarded by the network. The system state evolves as a series of discrete events, where the events of interest are arrival and departure of packets, and protocol timeouts.

Fluid flow: A fluid flow model sees the data transport as a continuous fluid, with no packet boundaries. State variables vary continuously, and are described using differential equations. In the context of congestion control, fluid flow models usually do not try to capture all details of the dynamics. Instead, the state variables represent averages of the true system state, where the average can be an average over an RTT, or an expected value with respect to stochastic features of the system.

Hybrid: In a hybrid model, the evolution of the state is a result of discrete events, together with continuous changes between events. Typically, a continuous model is used for the queueing dynamics and maybe for some of the end-host protocol actions, while protocol actions such as the multiplicative decrease in TCP is modeled as an discrete event, and a corresponding discontinuous state update.

In this section, we will describe these in turn. We will also describe the framework of network utility maximization, which is an important tool for analysis and understanding of large networks.

Packet-level models

For telecommunication networks, there is a long tradition of using queueing and traffic theory, going back to the work of Erlang in the early 20th century [41]. These are tools for analyzing quality measures such as queueing delay and blocking probability, given capacity and demand. The theory gives particularly nice results when the arrival of traffic is a Poisson process. A later development is adversarial queueing theory [21], where the arrival traffic is not stochastic, but chosen by an adversary, subject to load constraints.

Web-server admission control is studied in [99], with the central trade-off between request rejection rate, and response time of the admitted requests. The performance, in terms of blocking probability and delay, of a single server with several traffic classes, is analyzed in [4].

In [103], it is argued that traffic theory is an essential tool for developing an Internet with multiple service classes. Still, queueing and traffic theory is best suited for analysis of mechanisms related to quality-of-service and admission control, which are of an open-loop character, and more difficult to apply to closed-loop congestion control. A recent application to TCP/AQM-style congestion control is found in [53].

The properties of interconnected queueing systems, in particular when there are several service classes, is remarkably complex. A recent preprint [47] proves that for queueing networks with multiple service classes, a given deterministic arrival process, and given deterministic service times, stability is an undecidable problem. I.e., there exists no algorithm that takes as input a description of a queueing network, an arrival process, and an initial state, and determines whether or not the queue lengths of the system are bounded over time.

The network calculus framework is a development of queueing theory, working with bounds for the arrival rates and service rates [22, 36, 37]. Network calculus results give worst-case bounds for properties such as router queue sizes and end-to-end packet delay, which is useful for quality-of-service problems. However, the tools for analysis of dynamical systems within the network calculus framework is much less developed than the theory for dynamical systems described by ordinary differential equations. This makes it quite intricate to analyze feedback systems. A network calculus model for TCP is developed in [13], and it needs a fairly heavy

machinery, working with a large extended state, and exploiting TCP periodicity, in order to derive results for average properties.

When leaving the realm of analysis, and turning to simulation of queueing networks, the most widely used tool is the `ns2` network simulator, which keeps track of each packet as it traverses the network, as well as the protocol actions of communication end points [94]. When deriving theoretical results based on simplified network models, packet-level simulation as in `ns2` is the usual method for validating the results and its underlying assumptions.

Packet-level models naturally give an accurate description of the packet flow. There are powerful tools for analyzing expected values as well as worst-case bounds for properties such as the queueing delay and the probability of loss or blocking. A drawback is that it is difficult analyze feedback systems, where the arrival rates depend on the network state, in particular when we are interested not only in the equilibrium properties, but in the dynamics of the system.

Fluid-flow models

As the name implies, fluid-flow models do not try to describe individual packets, but view the data streams across the network as continuous flows. All the quantities of interest, such as sending rates, window sizes, and queue sizes, are treated as continuous (and often differentiable) functions of time. We start with the fluid-flow model for a queue.

Example 2.1 (Queue dynamics) Assume that we have packets arriving to a queue as a Poisson process with time-varying intensity $r(t)/m$, where $r(t)$ is continuous and m is the packet size. Also assume that the service times are exponentially distributed with parameter m/c, independent of the arrival process. This is an M/M/1 queue with arrival rate $r(t)/m$ and service rate c/m.

Let $N(t)$ denote the number of packets in the queue at time t. The amount of queued data is then $q(t) = mN(t)$. We now fix some t, with $q(t) > 0$, and examine the change of the queue size during the interval $[t, t+h]$. Let A_h be the number of arriving packets during this interval, then A_h is Poisson distributed with parameter $\int_t^{t+h} r(s)/m \, \mathrm{d}s$. Let S_h be the number of departing packets during the same interval. For small h, we can ignore the possibility that the queue becomes empty, and approximate S_h as Poisson distributed with parameter hc/m.

The change in the queue size is then

$$q(t+h) - q(t) = m\big(N(t+h) - N(t)\big) = mA_h - mS_h$$

From this expression we can easily compute the expected value, and, since A_h and

S_h are assumed independent, we can also compute the variance:

$$\mathrm{E}[q(t+h) - q(t)] = m\,\mathrm{E}\,A_h - m\,\mathrm{E}\,S_h = \int_t^{t+h} r(s)\,\mathrm{d}s - hc$$
$$\mathrm{Var}[q(t+h) - q(t)] = m^2\,\mathrm{Var}\,A_h + m^2\,\mathrm{Var}\,S_h = m\int_t^{t+h} r(s)\,\mathrm{d}s + hcm$$

If we keep h fixed, and let $m \to 0$, it follows that

$$q(t+h) - q(t) \to \int_t^{t+h} r(s)\,\mathrm{d}s - hc$$

in the L^2 sense. If we divide by h and let $h \to 0$, we find that

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = r(t) - c$$

When $q(t) = 0$, this equation must be extended with a state constraint,

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \begin{cases} r(t) - c & q(t) > 0 \\ \max(0, r(t) - c) & q(t) = 0 \end{cases}$$

This establishes the usual equation for queueing dynamics as the small packet limit of an M/M/1 queue. □

When modeling TCP, the fluid flow model is not motivated as a limit when the packet size tends to zero; it is unclear how to scale the protocol actions so that such a limit makes sense. Instead the fluid flow modelling tries to capture only the dynamics on a time scale larger than roughly one RTT. The sending rate in such a fluid flow model can usually be thought of as a moving average of the sending rate in the real system, where the average is taken over one RTT. When the system behavior depends on random events, such as mark bits set by some randomized algorithm, or packet drops that depend on the randomness in cross-traffic, there is another kind of averaging: The model attempts to track the *expected value* of the system state, which may be of different character than actual realizations.

Example 2.2 (TCP New Reno) We now consider the window dynamics of TCP New Reno. The state variable is the window size w, and the input signal is the packet loss probability p. Let m denote the packet size. Assume that the RTT is constant, and denote it by τ. For each RTT one full window of data is transmitted, consisting of w/m packets. This gives an average inter-packet time $T = \tau m/w$. For each packet, there are two possible window updates:

- With probability $1 - p$, an ACK is received, and the additive increase mechanism grows the window by $\Delta w = m^2/w$.

- With probability p, the packet is lost, and the multiplicative decrease mechanism reduces the window size by $\Delta w = -w/2$.

Next, compute the expected value of the window update,

$$\mathrm{E}[\Delta w] = \frac{(1-p)m^2}{w} - \frac{pw}{2}$$

We get the corresponding fluid flow differential equation by using the expression for the expected rate of change, $\mathrm{E}[\Delta w]/T$, as the derivative of a continuous function $w(t)$:

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{1}{T}\,\mathrm{E}[\Delta w] = \frac{(1-p)m}{\tau} - \frac{pw^2}{2\tau m}$$

□

To connect the fluid flow model for the queue (Example 2.1) with the above model for TCP, we need a relation between the window size and the sending rate. Since one window of data is transmitted per RTT, it is natural to define the fluid-flow rate as $r(t) = w(t)/\tau$. Then the source dynamics can be rewritten in terms of this rate, as

$$\frac{\mathrm{d}r}{\mathrm{d}t} = \frac{(1-p)m}{\tau^2} - \frac{pr^2}{2m}$$

This model is used in [80], and a related model, taking signalling delays into account, is used in [56]. The first term corresponds to the additive increase, and the second term corresponds to the multiplicative decrease. The equilibrium of this equation gives well known the TCP throughput formula

$$r = \frac{m}{\tau}\sqrt{\frac{2(1-p)}{p}} \tag{2.1}$$

If we substitute the relation $r(t) = w(t)/\tau$ into the queue dynamics, we get

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \frac{w(t)}{\tau} - c \tag{2.2}$$

This is a model for the dynamical system with the window size as input, and the queue size as output. If Eq. (2.2) is extended with the time-varying queueing delay, corresponding to $r(t) = w(t)/(\tau + q(t)/c)$, we get the *integrator link model*. This model, as well as the simpler Eq. 2.2, does not capture the effect of the ACK-clock. A different model, based on the ACK-clock, is the *static link model* $q(t) = w(t) - c\tau$ [114].

In Chapter 3, we describe and validate the new *joint link model*, with the following relation between window size and sending rate:

$$r(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \frac{\mathrm{d}w(t)}{\mathrm{d}t}$$

This new model unifies the two models, the integrator link model and the static link model, that are used in the literature.

The advantage of the fluid-flow approach is that we get models in terms of differential equations, and can use the powerful tools from control theory and dynamical systems. The main drawback is that the dynamics of the fluid-flow model does not capture system behavior on time scales shorter than one RTT, and in the case of systems with some stochastic components, we also lose information when we study the expected values only. Nevertheless, fluid-flow modeling, in combination with validation through experiments or `ns2` simulation, is the most widely used methodology in congestion control research.

Hybrid models

A *hybrid system* has both a discrete state, and some continuous state variables. Within each of the discrete states, the continuous state variables evolve according to some differential equations. Transitions between the discrete states can be triggered by conditions on the continuous state variables, or by external events, and each state transition may make discontinuous changes to the otherwise continuous state variables. One can think a hybrid system as a finite state machine, where each state is associated with its own dynamical system.

In congestion control, it is natural to model queue dynamics as well as the additive increase of the window as continuous processes, while queue overflow and the resulting packet drops can be modeled as a discrete event. A hybrid model for TCP, including not only congestion avoidance, but also the slow start phase, is described in [74]. In that paper, the hybrid model is used as a simulation tool, but it can also be used for analysis [19].

The AIMD style of congestion avoidance can be modelled as a hybrid system. There are discrete congestion events, which correspond to queue overflow, and result in discontinuous window updates. Between congestion events, window and queue sizes are modelled using fluid-flow differential equations. Such models are used in [14, 108], to analyze fairness properties, the statistics of end-to-end throughput, and rate of convergence.

Hybrid models are a compromise between packet-level models and fluid-flow models, both in terms of the accuracy of the model, and in terms of the difficulties in analysis of the models. In this thesis, Chapter 5 uses a hybrid model with discrete congestion events.

Network utility maximization

All the three modeling approaches described above, packet-level, fluid-flow, and hybrid modeling, are practical to use only for relatively small networks, or specialized topologies. It is difficult to analyze and understand the behavior for a large scale networks.

The framework of network utility maximization, pioneered in [65, 79], allows for the analysis of both static and dynamic properties in large networks with arbitrary topology.

A convex optimization problem Consider a general network consisting of L links, each of capacity $c_\ell > 0$, and S sources transmitting data with rate r_s. We describe the network topology using the *routing matrix* A, defined by

$$A_{\ell s} = \begin{cases} 1 & \text{if source } s \text{ uses link } \ell \\ 0 & \text{otherwise} \end{cases}$$

Note that a "source" here identifies not only the sending node, but the entire path through the network. A single node that transmits data to several destinations corresponds to several sources in this model.

The utility maximization problem associates a utility function $U_s(r_s)$ to each source, and tries to maximize the total utility, subject to the capacity constraints on the links. Let y_ℓ denote the aggregated traffic arriving at link ℓ, then $y_\ell = \sum_s A_{\ell s} r_s$. In vector form, this can be written as

$$y = Ar$$

The utility maximization problem can thus be stated as

$$\begin{aligned} \max_r \quad & \sum_s U_s(r_s) \\ \text{s.t.} \quad & Ar \le c \\ & r \ge 0 \end{aligned}$$

The capacity constraints define a convex set, with non-empty interior. If the utility functions U_s are concave, then this is a convex optimization problem with strong duality, and the optimum is characterized by the Karush-Kuhn-Tucker (KKT) conditions [23],

$$\begin{array}{lll} r \ge 0 & \lambda \ge 0 & \text{non-negativity} \\ y \le c & & \text{capacity constraint} \\ y = Ar & p = A^T\lambda & \text{aggregation} \\ p_s = U_s'(r_s) & & \text{gradient condition} \\ 0 = \lambda^T(c - y) & & \text{complementary slackness} \end{array}$$

Here, λ_ℓ is the dual variable or "price" associated with the constraint $y_\ell \le c_\ell$, and the aggregated price $p = A^T\lambda$ gives, for each source, the sum of the prices for the links used by that source.

Furthermore, if the utility functions U_s are *strictly* concave, the optimum rates are unique, and if in addition $\operatorname{rank} A = L$, then also the dual variables λ are unique.

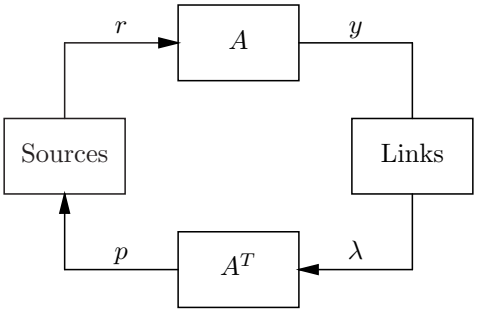

Figure 2.3: Distributed congestion control in the utility maximization framework. The "Sources" block on the left represents all the sources, where each source is a system with input p_s and output r_s. The "Links" block on the right represents all the links, where each link is a system with input y_ℓ and an output λ_ℓ. Links and sources are interconnected via the routing matrix A.

For the rank condition to hold, it is necessary (but not sufficient) that $S \geq L$, i.e., that there are more sources than link. If $S \geq L$, one way that the rank condition can fail to hold is if there are two links that are used by precisely the same sources. In this case, the individual prices for these two links are not uniquely determined, but their sum may still be.

Equilibrium properties The relevance of the utility maximization problem comes from the relation to the equilibrium of distributed congestion control. Consider the system in Fig. 2.3. Source s uses a sending rate r_s. Link ℓ sees an arrival rate y_ℓ, where the aggregation of source rates is given by $y = Ar$. Each link then determines its link price λ_ℓ. These prices are aggregated as $p = A^T\lambda$, and source s sees the path price p_s. In turn, this price influences the source's sending rate r_s.

Assume that the source dynamics, i.e., the relation between p_s and r_s, is described by the differential equation

$$\dot{r}_s = F_s(r_s, p_s)$$

Also assume that the link dynamics, relating λ_ℓ to y_ℓ, possibly via some internal state v_ℓ, is described by the equations

$$\dot{\lambda}_\ell = G_\ell(y_\ell, v_\ell, \lambda_\ell)$$
$$\dot{v}_\ell = H_\ell(y_\ell, v_\ell, \lambda_\ell)$$

Then, under certain conditions, the equilibrium conditions for this complex dynamical system is equivalent to a utility maximization problem with a certain utility function, where the λ_ℓ are the dual variables. This procedure is called "reverse engineering" in [30]. In the literature, it is for some reason usually described using

discrete time models, even though continuous time is more natural in the context of fluid flow modeling. For equilibrium properties, there is no fundamental difference between continuous and discrete time, though.

First, assume that

$$\begin{cases} G_\ell(y_\ell, v_\ell, \lambda_\ell) = 0 \\ H_\ell(y_\ell, v_\ell, \lambda_\ell) = 0 \end{cases} \implies \begin{cases} y_\ell \le c_\ell \\ \lambda_\ell(c_\ell - y_\ell) = 0 \end{cases} \tag{2.3}$$

Next, assume that $F_s(r_s, p_s) = 0$ can be solved for p_s,

$$p_s = f_s(r_s)$$

and that f_s is decreasing. Then the utility function defined by

$$U_s(r) = \int f_s(r)\,\mathrm{d}r$$

is concave. It follows that for any equilibrium r, λ, v of the dynamical system, r, λ are the optimum of the corresponding utility maximization problem. Unfortunately, this argument by itself does not establish the *existence* of an equilibrium for the dynamical system.

To establish existence of the equilibrium, a direct approach is to use a converse of (2.3). We then need the following additional assumption on G and H: For any $y_\ell, \lambda_\ell \ge 0$ such that $y_\ell \le c_\ell$ and $\lambda_\ell(c_\ell - y_\ell) = 0$, there exists some v_ℓ such that $G_\ell(y_\ell, v_\ell, \lambda_\ell) = 0$ and $H_\ell(y_\ell, v_\ell, \lambda_\ell) = 0$. A more general approach, hinted at in [30], is to apply Kakutani's fixed point theorem.

Discussion

In this thesis, fluid-flow models are the primary tools. This makes it possible to use the theory of dynamical systems and ordinary differential equations. In Chapter 5 we use a hybrid model, with congestion events corresponding to queue overflow and discontinuities in the window sizes, and between these events, queue and window sizes evolve continuously, according to the fluid-flow differential equations.

The utility maximization framework is used to get results and insights into the system behavior for large networks. Furthermore, packet-level `ns2` simulations are used for validation of theoretical results. Packet-level models, including queueing theory and network calculus, are not used in our analysis, since it seems quite difficult to apply these tools to feedback systems.

2.4 Related TCP-like protocols

The older members of the TCP family are TCP Tahoe, the first TCP implementation that included the congestion control mechanisms in [59], and TCP Reno, which introduced the fast retransmit and fast recovery mechanisms. The protocols are

Protocol	Type	Main objective(s)
TCP New Reno [45]	Loss-based	*(The standard TCP protocol)*
TCP Vegas [24]	Delay-based	Higher throughput and reduced loss rate.
Fast TCP [115]	Delay-based	Higher throughput with high capacity and large delays.
HSTCP [43]	Loss-based	”
STCP [66]	”	”
BIC-TCP [102]	”	”
CUBIC [102]	”	”
TCP Africa [69]	Loss-delay based	”
Compound TCP [112]	”	”
TCP Illinois [78]	”	”
Westwood+ [86]	Bandwidth estimation	Higher throughput over wireless networks. Also proposed for high capacity and large delay networks.
XCP [63]	Extra signalling	Higher throughput and faster convergence with high capacity and high delays. Smaller queues. Decoupling of congestion control and fairness control.

Table 2.1: Overview of proposed TCP-like protocols.

named after the corresponding releases of the Berkeley System Distribution (BSD), in 1988 and 1990, respectively.

Today, TCP New Reno is the most widely used protocol, officially elevated from Experimental status to a Proposed Standard in 2004 [45]. We will use TCP New Reno as the base protocol that newer protocols are compared to.

There are three main lines of the research that tries to improve TCP: Improving performance over links with large bandwidth-delay product, improving performance over wireless links, and reducing the queueing delay at bottleneck links, thereby improving quality for real-time applications.

Table 2.1 gives an overview of proposed TCP variants. Each protocol is classified by the control mechanism, or type of feedback, that it uses, and by the performance problem in TCP New Reno that it attempts to solve. In this section, we first discuss delay-based congestion control, followed by an overview of the protocols designed

for high speed, large delay networks. Outside this classification, we also describe the area of Active Queue Management (AQM), and rate-based congestion control guided by the utility maximization framework. Problems and solutions related to TCP over wireless are postponed until Sec. 2.7. Finally, the eXplicit Control Protocol (XCP, [63]) does not fit in any of the above groups. It introduces more precise signalling between senders and routers, and moves part of the congestion control algorithm from senders to routers.

Delay based congestion control

One line of TCP algorithms, of which TCP Vegas is the most well known, uses the queueing delay as a feedback signal for the window adjustment.

Proportional fairness To describe delay based congestion control, let τ_s^0 denote the roundtrip propagation delay of source s, τ_s the queuing delay over the entire path, and $r_s = w_s/(\tau_s^0 + \tau_s)$ be the equilibrium sending rate. The queueing delay τ_s is used as a feedback signal. For window-based control, it is shown in [89] that an equilibrium such that

$$w_s - r_s\tau_s^0 = \frac{p_s}{r_s^{\alpha-1}}$$

corresponds to (p, α)-proportionally fair rates. The left hand side represents the amount of in-flight data that is queued at some router along the path, and the right hand side can be thought of as a target value for the amount of queued data. When $\alpha > 1$, the right-hand side is state-dependent, and unbounded when r_s is small. So the case of $\alpha = 1$, corresponding to proportional fairness, is the one of most practical interest.

With this observation it is natural to use the difference $w_s - r_s\tau_s^0$ for a window update law. The control law

$$\dot{w}_s(t) = -\kappa\frac{\tau_s^0}{(\tau_s^0 + \tau_s(t))}\,\frac{w_s(t) - r_s\tau_s^0 - p_s}{w_s(t)} \tag{2.4}$$

is suggested in [89], where it is shown that this system is stable, and converges to a proportionally fair rate allocation. (In the proof, the relation between window sizes on one hand, and the sending rates and aggregate queueing delays on the other, is described by a static mapping. This can be seen as an implicit argument of time-scale separation).

To appreciate this result, consider a source sending data across the network, and trying to find its fair share of the capacity. Assume the flow traverses a single bottleneck, and consider two different cases for the competing traffic. In the first case, the bottleneck is shared with one large flow, and in the second case, the bottleneck is shared with a large number of small flows. In these two cases, the arrival rate and the queueing delay at the bottleneck router can be the same, although the fair share of the capacity is very different. So how can the flow distinguish between

these two cases, if it only measures the queueing delay? In the terminology of [89], the difference $w_s - r_s\tau_s^0$ gives a "decoupled fairness criteria". By monitoring this value, and adjusting the window size so that the difference converges to, say, three packets, the rates will converge to a proportional fair sharing of capacity.

TCP Vegas and Fast TCP The design objectives of TCP Vegas were to improve throughput and reduce packet losses for TCP connections across the Internet [24]. It uses the control law

$$\dot{w}_s(t) = \begin{cases} \frac{m_s}{\tau_s^0+\tau_s} & w_s(t) - \frac{w_s\tau_s^0}{\tau_s^0+\tau_s} < \alpha \\ -\frac{m_s}{\tau_s^0+\tau_s} & w_s(t) - \frac{w_s\tau_s^0}{\tau_s^0+\tau_s} > \beta \end{cases}$$

where the parameter m_s is the packet size (resulting in the same linear increase as TCP Reno in the absence of congestion), and typical values for the two thresholds α and β are one and three packets, respectively. For a recent stability result for this highly non-linear feedback law, see [32]. (TCP Vegas also introduces new mechanisms for the slow start and for loss detection and recovery, which will not be discussed further here).

Fast TCP is a design that is inspired by TCP Vegas, but is aimed for solving TCP's performance problems over connections with high capacity and long delays, and hence with a large bandwidth delay product [115]. The window update law is expressed in discrete time, and can be written as

$$w_s(t+1) = w_s(t) + \gamma_s(\alpha_s - r_s(t)\tau_s(t))$$

If we note that $r_s(t)\tau_s(t)$, the amount of queued data, is the same as $w_s(t) - r_s(t)\tau_s^0$, this looks like a discrete time variant of the control law (2.4), with a slightly simpler expression for the gain.

Estimating the propagation delay Delay based congestion control is attractive, because it can achieve fairness, with no other signals than the queueing delay, which is provided by the network for free. To implement delay-based congestion control, it is essential to measure the propagation delay τ_s^0. In practice, each source usually sets this to the smallest observed RTT. This is accurate if the propagation delay is constant and the router buffer is empty when the source starts sending packets.

However, if for example one flow arrive after another, each flow tries to keep three packets in the bottleneck queue, and each flow uses the smallest observed RTT as its estimate of the propagation delay, then the queue will build up. Flows arriving later will get a larger estimate of the propagation delay than earlier flows, and hence more capacity than their fair share. This effect is described in [81]. The discrepancy between real propagation delay and the estimate can be even more pronounced in case a flow with a delay-based congestion control share a bottleneck with TCP New Reno, which does not even try to keep the queue size down. For an

illustration of this effect, see Chapter 5, where the window size and throughput of TCP Westwood+ depends on its estimation of the propagation delay, even though it is not a purely delay-based congestion mechanism.

To alleviate the problems with estimation of the propagation delay, one method, also described in [81], is to combine TCP Vegas with explicit feedback from an AQM mechanism in the network, in this case Random Exponential Marking. If this system is tuned so that the queue stays almost empty, all flows can get an accurate estimate of the propagation delay. This solution to the problems comes with the cost of reduced link utilization; on the order of 5–10% of the capacity is unused. As is typical for AQM mechanisms, this system also works better as the number of flows grow.

Estimating the queueing delay is even more challenging if the RTT contains more components than a fixed propagation delay and time varying queueing delay, e.g., retransmission delays from wireless link or a highly loaded Ethernet link, or changes to the propagation delay due to mobility and other topology changes. There is little research done on the topic of delay-based congestion control in the setting of wireless mobile network. An exception is the study [62], which constructs a delay based congestion control scheme for commercial CDMA (Code Division Multiple Access) network. In this network, the measured delays include a random component, and in addition, the packet loss probability is several percent.

High speed, large delay networks

To improve TCP performance when the bandwidth-delay product is large, it is natural to just make TCP's increase rule more aggressive. Experimental protocols include High-Speed TCP (HSTCP, [43]) and Scalable TCP (STCP, [66]). However, this leads to fairness problems when links are shared with TCP Reno.

> *It is interesting to note that the increase rate for* HSTCP *and* STCP *grows as the window grows. As a result, these protocols are in fact most aggressive just at the moment where they are sending at maximum capacity. Intuitively, this is the time when a protocol should be the least aggressive.*
>
> R. King et al., [69]

The BIC-TCP algorithm and its successor CUBIC avoids this problem by keeping track of $w_{\max}$, the window size where it previously experienced a packet loss, and slowing down the window increase as the window size approaches $w_{\max}$ [102]. CUBIC is implemented in the Linux kernel since version 2.6.16.

Loss-based congestion control A different approach to back down from an aggressive window increase rule when the network approaches congestion are the so called *loss-delay based* congestion control algorithms. Like CUBIC, these protocols try to increase the window size more aggressively than TCP Reno as long as the network is underutilized, and switch to a behavior similar to Reno's AIMD when the

network is close to congestion. Delay measurements are used to select the mode of operation.

TCP Africa uses a slow mode that is identical to TCP Reno, and a fast mode with an increase rule borrowed from HSTCP. To switch between modes, the number of queued packets is estimated in the same way as in TCP Vegas, and slow mode is selected when the number of queued packets exceed a threshold [69].

Compound TCP does not have such a discrete switch between modes. Instead, it uses a rate based on the sum of the congestion window and a *delay window*, where the latter is increased rapidly when the network is underutilized [112]. The multiplicative decrease when a packet loss is detected, is the same as in TCP Reno. Compound TCP is implemented in Microsoft Windows Vista.

TCP Illinois also has no discrete mode switches. It uses AIMD, but with parameters that depend on the delay [78]. The window update is parameterized as

$$w(t+1) = \begin{cases} w + \alpha/w & \text{for each ACK} \\ w - \beta w & \text{for each packet loss} \end{cases}$$

The additive increase parameter α and the multiplicative decrease parameter β are continuous functions of the RTT. The functions are chosen so that when the RTT is small, interpreted as low congestion, α is large and β is small, and when the RTT is large, interpreted as a network close to congestion, α is small and β is large.

Active queue management

Active queue management (AQM) refers to control that takes place in network routers. The input to the control is the arrival rate and queue size for a particular outgoing link, and the output is a decision on how to mark or drop packets. Usually the output is applied in a stochastic way; the AQM mechanism selects the probabilities for marking and dropping packets, and then each forwarded packet is marked or dropped with the given probability. End hosts will then react to these signals and adjust their sending rates.

It is natural to think about congestion control as controllers located at the end hosts, controlling the system of interconnected queues. The sending rates (or window sizes) are the control signals, the queue sizes in intermediate routers are the system's state, and the loss probability and the delay are the measured outputs. In a way, AQM is the dual approach: The process being controlled is the collection of end hosts. The control signals are the probabilities for marking and dropping of packets, and the arrival rates and queue sizes are the measured outputs.

Explicit congestion notification It is preferable if the AQM mechanism can mark rather than discard selected packets. Packet marking is known as Explicit Congestion Notification (ECN). Using ECN requires support in end hosts, which are required to signal that they are ECN-capable, and to react to set marks in the same

was as to an actual packet loss. ECN uses bits in the IP packet header, and its use is specified in [101].

There are several motivations for using AQM, corresponding to difference aspects of quality and performance.

- Reduced packet loss rate.
- Reduced queueing delay and jitter.
- Reduced synchronization between the AIMD window fluctuations of different flows.
- Improved throughput.

In [73], several AQM mechanisms are evaluated, with and without the use of ECN. The quality measure of this study is user response time for web browsing. The main results were summarized as follows:

> 1. *For offered loads up to 80% of bottleneck link capacity, no* AQM *scheme provides better response times than simple drop-tail* FIFO *queue management.*
> 2. *For loads of 90% of link capacity or greater when* ECN *is not used,* PI *results in a modest improvement over drop-tail and the other* AQM *schemes.*
> 3. *With* ECN*, both* PI *and* REM *provide significant response time improvement at offered loads above 90% of link capacity. Moreover, at a load of 90%* PI *and* REM *with* ECN *provide response times competitive to that achieved on an unloaded network.*
> 4. ARED *with recommended parameter settings consistently resulted in the poorest response times which was unimproved by the addition of* ECN*.*
>
> Le et al. [73]

Here, REM stands for Random Exponential Marking [10], PI is Proportional Integral AQM [57], and ARED is Adaptive Random Early Detection [44]. We can conclude that AQM matters only for links that are highly loaded, and that ECN is essential to give any significant improvement in terms of user response time.

Most AQM controllers measure the queue length, and use this value as the input to the controller. It is also possible to use the arrival rate. The control scheme in [110] uses proportional rate control together with proportional integral control of the queue length. The relation between rate-based and queue-based AQM mechanisms is investigated in [109], in the limit of a large number of flows.

Being a dual approach, AQM design usually considers the dynamics at senders to be given, usually as AIMD-style control like with TCP Reno. Many different control strategies have been tried, e.g, the proportional integral controller. One challenge is

the severe uncertainty about important parameters such as the number of flows, and the RTT and window sizes of the end-to-end flows. This has led to interest in robust control design for TCP/AQM [118, 83, 120], as well as fuzzy control [33, 8, 111].

Rate based congestion control

Much of the recent work on congestion control uses the network utility maximization framework as guidance for design and analysis, as pioneered in [65, 79]. Proposed schemes can be classified as

Primal: At the links, the price is a static function of the aggregate arrival rate. Sending rates at sources are adjusted dynamically.

Dual: At the links, there is a dynamic relationship between the aggregate arrival rate and the link price. E.g., if the queueing delay is used as the link price, this is the usual queueing dynamics. At the sources, the sending rates are determined as a static function of the feedback, i.e., the "aggregate price" provided by the network.

Primal-dual: These schemes use dynamic control laws at both sources and links.

In the dual algorithm, there is no state at the sources, while for the primal and the primal-dual algorithms, the state variable at each source is the sending rate. That the sending rate is controlled directly means that these algorithms do not use the usual notion of a *window size*. Stability is a central issue. Stability results can be local or global, include or ignore signalling delays, and be valid for special topologies or for arbitrary network topologies.

Stability of primal mechanisms is analyzed in [72, 119], with a focus on the effect of communication delays.

A dual algorithm is proposed in [95], where it is shown to be locally stable for arbitrary topologies and delays. For the single-link single-source topology, [96] gives a proof of global stability, and an improved stability condition on the source gain.

A primal-dual algorithm is studied in [6], which proves stability for an arbitrary topology in the absence of delay, and for the topology with a single bottleneck link and several users, where each user can have a different RTT.

A general stability theory for linear systems on a graph is developed in [77]. One of the applications is rate-based congestion control.

Discussion

In this thesis, Chapter 6 considers the design of end-to-end congestion control. The proposed protocol is close to the TCP/AQM class of proposals, in that it lets routers mark packets with a marking probability that depends on the queue size. However, the response to the feedback from the network is additive, like in XCP, rather than multiplicative as in the usual handling of ECN. The protocol is also related to delay-based congestion control, in that queue sizes are the primary variables that we want

to control. One important difference is that we have a per-link tuning knob. These similarities and differences are discussed further in Sec. 6.3.

2.5 Fairness

Fairness is one of the central equilibrium properties of any congestion control scheme. It is also one of the control objectives. To quote [89], "Roughly, a fair scheme is one that does not penalize some users arbitrarily.". There are several possible notions of fairness

Max-min fairness A max-min-fair rate vector r^* is defined as follows. The vector must be feasible, $r^* \geq 0$ and $Ar^* \leq c$, and have the property that if $r_s > r_s^*$ for some other feasible rate vector r and some source s, then there is some other source s' such that $r_{s'}^* \leq r_s^*$ and $r_{s'} < r_{s'}^*$. That is, no rate can be increased, without simultaneously decreasing some other rate which already is smaller.

Proportional fairness A rate vector r^* is proportional fair if it is feasible, and if for any other feasible rate vector r it holds that

$$\sum_s \frac{r_s - r_s^*}{r_s^*} \leq 0$$

α-fairness Proportional fairness and other fairness notions are generalized as follows. Let p be a vector of weights, $p_s > 0$, and let $\alpha \geq 1$ be a scalar parameter. A feasible rate vector r^* is (p, α)-proportional fair, if, for any other feasible rate vector r, it holds that

$$\sum_s p_s \frac{r_s - r_s^*}{(r_s^*)^\alpha} \leq 0$$

In this definition, $\alpha = 1$ corresponds to proportional fairness.

There is a nice connection between the notion of α-fairness and the utility maximization problem. First define a family of scalar, concave functions,

$$f_\alpha(r) = \begin{cases} \log r & \alpha = 1 \\ -r^{-(\alpha-1)} & \alpha > 1 \end{cases}$$

Then a rate vector r^* is (p, α)-proportional fair if and only if it maximizes the utility function

$$U(r) = \sum_s p_s f_\alpha(r_s)$$

Furthermore, in the limit as $\alpha \to \infty$, the maximizer converges to a max-min-fair rate vector. This correspondence is proved and discussed further in [89].

If one analyzes a congestion control protocol, using the "reverse engineering" procedure to find a utility function, the connection between fairness and the shape of the utility function helps us understand the fairness properties of the protocol. It has been shown that TCP Reno corresponds to $\alpha = 2$, while delay based congestion control mechanism such as TCP Vegas and Fast TCP corresponds to $\alpha = 1$, i.e., proportional fairness [30].

Problems with the notion of fairness

Of the design objectives for congestion control, fairness is the one that is hardest to define in a precise and universally accepted way. E.g., TCP New Reno gives rates that are inverse proportional to the RTT. This can be seen as an example of an arbitrary penalty on users connected with long-delay links. Or it can be seen as a reasonable weighting, viewing the propagation delay as a crude measure of the amount of networking resources that the user is claiming a share of.

Security One may think of fairness as a security measure that is intended to prevent a malicious user from using too much of shared resources, and thus harming other users who also need their share. From this point of view, leaving enforcement of fairness to end-to-end congestion control makes no sense.

> *Building a trusted client in software, and trying to limit the abilities of a user, on a general purpose computer is doomed to failure. For now, though, it provides a nice false sense of security.*
>
> Bruce Schneier [107]

There is nothing preventing a user from tuning his TCP implementation to be more aggressive than the standard version, and gain a larger than fair share. It is even conceivable that operating system vendors could start a war of tweaking TCP parameters, trying to attract customers to the operating system that gives them the highest performance (at the expense of users of other operating systems).

Flow-aware networking To move responsibility for fairness from the end hosts to the networking seems to be against the spirit of the end-to-end principle, in that it requires core routers to keep track of flows and keep per flow state. The amount of state needed is however smaller than one might expect. In [71], it is noted that to implement per-flow fair queueing, the scheduler need only handle the large flows. And for any link, the number of large flows is limited, which bounds the state needed for the scheduler. In addition, one bit per flow is needed to distinguish between large and small flows, and a couple of more bits in order to also support admission control.

Flow rates vs congestion cost When implementing fairness, the ultimate objective is fairness between real users. But there is no simple correspondence between

users and flows, since a user may run several applications each generating several flows.

> *We expect to be fair to people, groups of people, institutions, companies—things the security community would call 'principals'. But a flow is merely an information transfer between two applications. Where does the argument come from that information transfers should have equal rights? It's equivalent to claiming food rations are fair because the boxes are all the same size, irrespective of how many boxes each person gets or how often they get them.* B. Briscoe [25]

In [25], it is argued that defining fairness in terms of flow rates is a fundamentally flawed idea. Instead, fairness should be defined by quantifying the congestion cost each real user or organization causes for other users of the network, integrated over time. The notion of congestion cost goes back to Kelly's application of proportional fairness to the rate per unit charge [65]. The use of ECN-like marking of packets makes it possible to measure the congestion cost where traffic crosses an administrative boundary, and then it is possible (although not mandatory) to charge for the amount of congestion caused. The market price for congestion would then correspond to the marginal cost of upgrading network capacity.

Discussion

Chapter 5, we analyze the fairness between two different TCP versions, TCP New Reno and TCP Westwood+. When introducing a new version of a protocol in the Internet, it is important to understand the consequences for users and applications using the older version.

In Chapter 6, we use the utility maximization framework and the reverse engineering procedure to understand how resources will be shared. Concluding that the equilibrium of a distributed congestion control algorithm is the unique maximum of a maximization problem, with a certain utility function, makes the sharing of resources predictable and easier to understand. And it also makes it easier to compare the sharing produced by different protocols. In this way, fairness, and the related machinery of network utility maximization, is a means rather than an end.

2.6 Reducing queueing delay

Small queueing delays are important for the quality of real-time applications. Since the AIMD control in TCP makes bottleneck queues grow until they overflow, the quality of real-time applications is degraded when they share a bottleneck with TCP traffic. For this reason, we consider reduced queueing delays as one of the control objectives for congestion control. In this section, we describe router buffer sizing, which has a direct influence on queueing delays, and we give an introduction to the field of quality of service. Quality of service mechanisms tries to divide the traffic

into separate classes, depending on application requirements, and treat packets differently depending on their class. E.g., voice over IP could be given priority over web browsing which in turn is given priority over batch downloads.

Router buffer sizing

There is a sharp distinction between the transmission queues for bottleneck links and the queues for non-bottleneck links in the current Internet, with mostly drop-tail queues. The queue for a non-bottleneck link is naturally almost empty. The queue for a bottleneck link (or *bottleneck queue*, for short) on the other hand, fills up completely. This is a consequence of the simple drop-tail queueing discipline and the congestion avoidance mechanism of TCP. It is also important to note that the maximum queue capacity, i.e., the size of a full queue, is usually quite large. The rule of thumb, documented in RFC 3439 [26], is to use 250 ms times the link capacity, where 250 ms is a reasonable upper bound on end-to-end RTT on the Internet. This recommendation implies that the delay across a path in the network is the propagation delay plus 250 ms for each bottleneck link on the path. Large delays are undesirable for several reasons, including degraded quality for real-time traffic (e.g., voice over IP) and interactive traffic (e.g., telnet sessions and multi-player games). Large delays in the core network are particularly undesirable, which is one reason why most of the backbone links in the Internet are highly over-provisioned.

For the motivation of the rule of thumb, consider a single bottleneck with capacity c and buffer size b, and a single TCP flow with roundtrip propagation delay τ, divided into forward delay τ_{f} between sender and bottleneck, and backward delay τ_{b} from bottleneck to receiver, and back to the sender. When the buffer is full and the bottleneck queue starts dropping packets, the TCP sender reduces its window size, i.e., the amount of in-flight data, by half. Of the in-flight data, b packets reside in the router buffer, and $c\tau$ are in transit over some of the links. For the links to maintain full utilization, the window reduction should correspond to parts or all of the *buffered* packets, but leave $c\tau$ packets in the network, $c\tau_{\mathrm{b}}$ in transit after the bottleneck, and $c\tau_{\mathrm{f}}$ in the buffer (to compensate for the lack of packets in in transit on the links between the sender and the bottleneck). Hence $b \geq c\tau$ is required for full utilization.

In recent years, the rule of thumb has been questioned [11, 9, 38, 51, 12, 48]. When a large number of flows share a bottleneck, and the flows are not synchronized (meaning that they do not all reduce their window sizes at the same time), less buffering is required for full utilization. And furthermore, the loss in utilization is modest, even for buffers significantly smaller than suggested by the rule of thumb. For the single flow, single bottleneck scenario, with $b = c\tau/10$, one tenth of the rule-of-thumb size, the utilization is still roughly 75% [12].

Quality of service for real-time applications

A *real-time* network application can be defined roughly as an application whose quality degrades significantly with network delay. This links real-time to *interactivity*. Common real-time applications are telephone calls (Voice over IP), video conferencing, and network games. Even an `ssh` session could be classified as a real-time application, since it is annoying to the user if the delay from typing a character until it is echoed back is noticeable.

Whenever a delay of one or a few seconds is acceptable, the application usually has no real-time requirements on the network. E.g., most cases of streaming video have no fundamental real-time requirements, since a delay of one second or two at the start, used to fill up a large receiver-side play-out buffer, is acceptable. With a high capacity network, a one second delay will be sufficient to transfer ten seconds or more of video, and this amount of buffering should be more than sufficient to compensate for both jitter and occasional packet losses. Interactive video conferencing, and real-time coverage of sports and other events, are the exceptions.

Internet Quality of Service is a large research area, where one of the main objectives is coexistence of real-time applications and other network applications, all sharing the same networking infrastructure. Approaches include priority marks on packets (Differentiated services [93]), end-to-end reservation of resources with some quality guarantee (Integrated services [116]), and reservation of capacity for traffic aggregates using Multi Protocol Label Switching (MPLS [104]). The latter approach is in wide use, usually for managing traffic within the single administrative domain of an Internet Service Provider, and sometimes used for Virtual Private Network services to their customers, but it is only indirectly addressing end-to-end quality. An overview of quality-of-service mechanisms is given in [121].

It remains a very difficult problem to provide end-to-end quality guarantees for traffic over arbitrary networking paths, which uses links and routers belonging to many different administrative domains.

Two important questions are:

- Are queueing delay or congestion packet losses, in the current or near-future Internet, a problem for the quality of real-time applications?
- If so, where in the network are the problematic bottlenecks located?

The answers will naturally depend on which application and which parts of the network are investigated. One study of Voice over IP quality between academic sites in Europe and the Americas found that quality was generally good, compared to telecom standards, with some problems for the connections to the Argentinian and the Turkish measurement site [84]. The average values were 136 ms and 4.1 ms for the delay and jitter, with a loss probability of 1.8%.

Links in the backbone network are usually over-provisioned, and backbone capacity is getting less expensive. For example, for the Nordic university network (NORDUnet), the cost for backbone capacity and transatlantic lines were more then

90% of the budget in the late 1990s, reaching an all time high of 20.7 million Euro in 2000, with a transatlantic capacity of 777 Mbit/s. But since then, the cost has fallen each year, even though the capacity has continued to grow [75].

When the backbone is over-provisioned, the typical bottleneck links are a user's ADSL line, or the connection between a company's high-speed internal network and the rest of the Internet. For these bottlenecks, quality of service can be provided by manually giving higher priority for certain real-time applications, or by imposing capacity limits or traffic shaping for file transfers.

For example, consider Alice and Bob sharing an apartment and an ADSL line. Alice uses a peer-to-peer file-sharing application, such as bittorrent, to transfer large files over TCP. At the same time, Bob plays Counter Strike, which uses UDP. In this scenario, their ADSL router is a bottleneck, and its queue will fill up, increasing the RTT for all connections by up to several 100 ms, reducing Bob's chances of survival considerably. By configuring their router to simply give priority to UDP traffic, or by introducing an artificial bottleneck between Alice's computer and the ADSL router, at 90% of the capacity, this quality of service problem is solved.

Discussion

The congestion control protocols designed in this thesis, in Chapter 6 and 7, aims to maintain small queueing delays, and low packet loss rates, as a part of the congestion control. To do this, sources need some congestion information from the network besides packet losses; when a packet is lost due to queue overflow, it too late to prevent the queue from getting large. In Chapter 6, we use congestion feedback similar to that with AQM and ECN, while in Chapter 7, we use explicit cross-layer signalling from the link-layer at the bottleneck.

2.7 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 2.4.

Today, the mobile telephony system is now slowly merging 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. Ideally, the cellular radio network should just be yet another link-type for IP transmission, where the details are taken care of in the link-layer; upper layers should not need to know or care. However, that is easier said than done, because the wireless link available to the terminal has very 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 2.5. On the sending side, at the left in the figure,

Figure 2.4: 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.

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 (channel fading) and an additive noise, g and n in the figure. These are time-varying due to mobility of antennas and obstacles, and due to the 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)[2] 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 at which time, as well as

[2]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 CDMA systems, interference usually dominates over thermal noise.

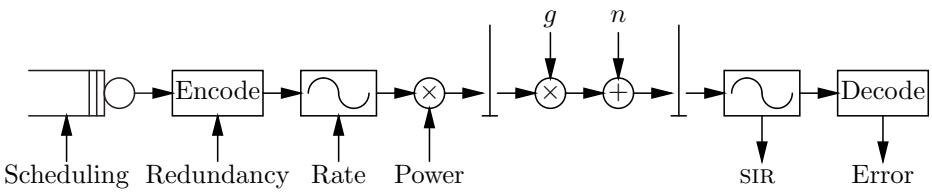

Figure 2.5: Wireless link mechanisms. From left to right: Buffer of frames to send, channel coder, modulator, transmission amplifier, transmission antenna, channel gain, channel noise, reception antenna, demodulator, and channel decoder.

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 Chapter 7, the focus is on cellular architecture, where base stations are distributed geographically, and where each base station handles a number of mobile terminals.

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. In 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.

Wireless link characteristics

When a particular link is used for IP transport, the three most important characteristics of the link are the capacity, the delay, and the loss probability. For wired links, these are constant, and the loss probability is usually very small. For wireless links, all three are time-varying, and the statistics depends on the trade-offs made in the link-layer design. E.g., the introduction of link-layer retransmissions reduces the loss probability, and pays the price of larger delay and larger delay variations, while adaptation of the parameters for modulation and forward error correction can trade capacity for a reduced loss probability.

Communication end-nodes may have some direct knowledge of the link they are directly attached to, but they primarily observe the characteristics of the entire network path they are using. With no cross-traffic, the observed capacity would be the capacity of the smallest link on the path, but with cross-traffic, the observed capacity depends not only on the links on the path, but also on the cross-traffic, which in turn depends on other parts of the network topology. The observed delay is the sum of the delay of each traversed link, and the queuing delay. The observed loss rate is the combination of the loss rate at each link (links can usually be considered independent), and congestion losses at any bottlenecks along the path.

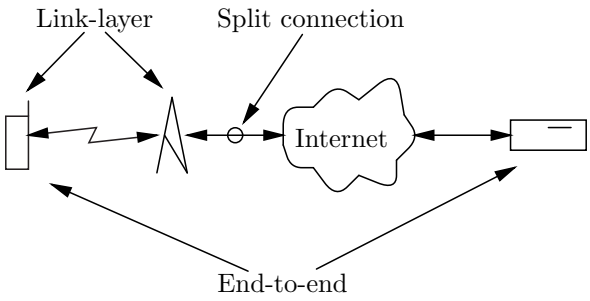

Figure 2.6: Approaches to the TCP over wireless problem.

Application requirements can be formulated in terms of capacity, delay and loss, for the network path they are using. E.g., for download of large files using TCP, a small loss probability is of highest importance, followed by high capacity, while delay is secondary. For an interactive TCP connection, such as a `telnet` or `ssh` session, small loss rate and small delay is important, but not capacity. And finally, for a real-time video conference application, small delays are of primary importance, and the application can make its own coding tradeoffs based on the available capacity and the actual packet loss rate.

TCP over wireless

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 research 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.

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

Cross-layer: Introduce extra signalling, beyond what is provided in the layered architecture. E.g., signalling from the link-layer at a wireless link, to the transport layer in a communication end point.

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.

These approaches are illustrated in Fig. 2.6. The mechanism described in Chapter 7 falls into the split-connection and cross-layer classes.

End-to-end mechanisms 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 time-stamp option) to make it possible for the sender to distinguish between acknowledgements for original transmissions and for retransmissions [82]. 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 and Westwood+ is to estimate the available bandwidth over the path, based on the timing of received ACKs [86, 52]. In the absence of packet losses, Westwood uses the same additive increase as TCP New Reno. But when a loss is detected, Westwood does not halve its window size, instead it sets the window size to the estimate of the available bandwidth times the smallest RTT it has observed so far. The rationale for this choice of window size is that it is sufficiently small to allow queues on the path to drain, but not smaller.

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 [28, 106, 46]. 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 mechanisms The link-layer approach tries to model the underlying radio channel. This model is then used 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 [3]. In [68], 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 Wide-band Code Division Multiple Access (WCDMA) system with joint rate and power adaptation [58], the tradeoff between link-layer Forward Error Correction (FEC) and TCP throughput [31, 16], and the tradeoffs between FEC, link-layer retransmissions, and transmission power [29, 17].

A hop-by-hop congestion control mechanism is designed and analyzed in [117], where congestion information is propagated "upstream", in the reverse direction of the data flow.

Sometimes, the interaction between link-layer dynamics and TCP gives counter-intuitive results, e.g., in certain circumstances TCP performance can be improved by artificially inflating the delay at the wireless link [70, 91].

Split-connection Compared to the problem of improved control at the link-layer and end-to-end, split-connection is conceptually simple. Instead if a single TCP connection between end points, the terminal connects to a proxy in the operator's network (e.g., for web browsing and file download, this could be an ordinary HTTP proxy). The proxy connects to the true end node, and forwards data from one connection to the other.

Another variant of the split-connection approach 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 [20]. 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.

Berkeley Snoop is a well known example of a performance enhancing proxy, which tries to maintain end-to-end TCP semantics [15]. It is an agent deployed at the router in front of a wireless link. The Snoop agent caches unacknowledged TCP segments and retransmits lost segments locally. Furthermore, it suppresses duplicate acknowledgements, to avoid triggering the fast retransmit mechanism at the TCP sender.

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 or even instability.

In [100], it is demonstrated by examples that wireless effects such as interference between nearby links can make TCP unstable. The congestion control system can have multiple equilibria, and the equilibrium that corresponds to fair shares may be unstable.

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 7. 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, in particular in combination with a split-connection proxy, provides 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 [27, 64].

2.8 Summary

In this chapter we have reviewed the Internet architecture, the workings of the TCP protocol, and the mathematical tools used for analysis of congestion control. We have made an excursion into the jungle of proposed improvements to TCP, for both wired and wireless networks, and we have looked closer at two of the control objectives: fairness and small queueing delays. After these preparations, we are ready for the modeling, analysis and design of window-based congestion control. In the next chapter, we start with modeling of the mechanism that forms the inner-loop of all window-based congestion control protocols.

Chapter 3

An accurate model for window-based transmission

> *These models do not adequately capture the self-clocking effect where a packet is sent only when an old one is acknowledged, except briefly and immediately after the congestion window is changed. This automatically constrains the input rate at a link to the link capacity, after a brief transient, no matter how large the congestion windows are set.*
>
> Wang et al. [114]

WINDOW-BASED congestion control can be seen as cascaded control system with two feedback loops. The inner-loop determines when each packet is transmitted, and hence the sending rate. The feedback signal is the arrival of ACKs, and the window size is a controller input. The outer-loop adjusts the window size, based on the received ACKs and any other available feedback information from the network. These two loops are illustrated in Fig. 3.1. In this and the next chapter, the focus is on the behavior of the inner-loop. We view the window sizes at the senders as the system inputs, and the queue sizes at network routers as the system outputs.

3.1 Overview

The inner-loop determines the dynamic relations between window size, sending rate, and queue size. Having a good model for the relation between the window size and the sending rate (which are coupled via the queueing state in the network) is motivated by two needs:

- For modeling of the network dynamics, one must fit together the rate-centric models for the queueing network with the window-centric control laws of TCP. To do this, the relation between window size and sending rate is an essential link.

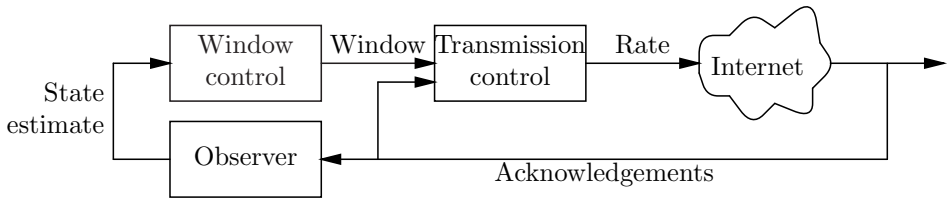

Figure 3.1: Window-based congestion control viewed as a cascaded control system: an inner-loop based on the ACK-clock and an outer-loop governed by a window controller.

- When designing the control laws for the outer-loop, it is essential to understand the dynamic properties of the inner-loop, in particular its stability properties, and the dominating time constants.

In this chapter we will develop and validate such a model, and the queueing dynamics it implies. The next chapter will examine the stability properties of this inner-loop. We will see that the amount of cross-traffic is an important system parameter, where different levels of cross-traffic give significantly different dynamical properties, in terms of the static gain and the dominating time constant. The main novelty in our model, is that it captures this influence, and gives accurate predictions for any level of cross-traffic.

This chapter is organized as follows. Sec. 3.2 describes the underlying model of the network, and the simplifications and limitations of that model. Sec. 3.3 investigates the relation between window size and sending rate. We describe two models, the integrator model and the static model, that are used in the literature. The new joint link model is introduced in Sec. 3.4. By linearization, we investigate how key system properties depend on the level of cross-traffic, and how the joint model differs from the earlier models. In Sec. 3.5, we describe validation experiments for the most basic topology of a single bottleneck link, one flow using with window-based congestion control, and some cross-traffic that is not subject to any congestion control. In Sec. 3.6, the model is extended to several flows, with different roundtrip propagation delays. Finally, in Sec. 3.7, we consider an arbitrary network topology, and show that the window-based transmission control has a unique equilibrium.

3.2 The network model

Topology

We model the network as a directed graph, with nodes, representing devices attached to the network, and directed edges, representing the communication links. An example is given in Fig. 3.2. This may seem like an obvious and accurate model, but it actually ignores that many links are not point-to-point links. Many link types, in particular wireless networking, but also wired Ethernet, are based on a shared

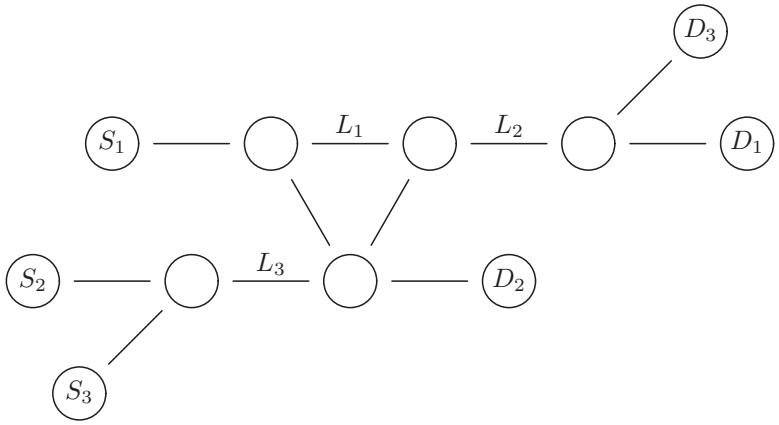

Figure 3.2: An example network graph. In this example, each edge represents two links, one in each direction. The source node S_k is transmitting data to the destination node D_k. The five intermediate nodes are routers. The links marked L_j are potential bottlenecks, in the direction from left to right.

communications medium. Several nodes are connected to the same link, and any node can transmit to any other node, where the sharing is controlled by the medium access control. E.g, if two hosts and an access point are connected with a 11 Mbit/s wireless Ethernet, the actual capacity constraint are quite different than if each host would have separate up- and down-links to the access point, of 2.75 Mbit/s each.

This graph model is accurate when the traffic on all bottleneck links are dominated by a single sender and a single receiver. E.g., in the wireless Ethernet example, if one host downloads a file over wireless Ethernet, and the uplink traffic from that host, as well as the uplink and downlink traffic for the other host, is small, then it is a reasonable approximation to model the link as a ≈ 11 Mbit/s directed point to point link from the access point to the downloading host.

Another approach to handle shared links is to focus on a pair of sender and receiver nodes, and model all traffic generated by other nodes as contributing to a time-varying capacity of the channel between the sender and receiver pair of interest.

Queueing

In the graph model, there is only a single node sending on each link. Each link ℓ has a finite, possibly time-varying, capacity c_ℓ. When the arrival rate to the link, i.e., the proportion of the total arrival rate to the router serving link ℓ, which are routed to link ℓ, exceeds capacity, the excess arrivals are queued. We denote the arrival rate as y_ℓ, and the queue size as q_ℓ. The queueing dynamics is an integrator,

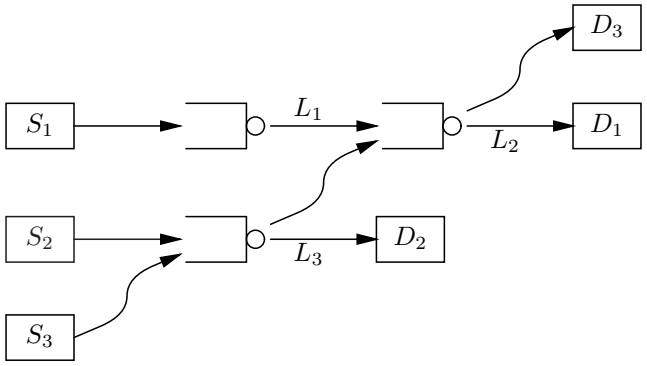

Figure 3.3: Example topology with three flows and three bottleneck links.

with a state constraint to ensure that the queue size stays non-negative at all times.

$$\dot{q}_\ell(t) = \begin{cases} y_\ell(t) - c_\ell & q_\ell(t) > 0 \\ \max(0, y_\ell(t) - c_\ell) & q_\ell(t) = 0 \end{cases}$$

The queueing delay, affecting a packet arriving at time t, is $q_\ell(t)/c_\ell$ (this simple relation between queue size and delay breaks down if capacity is not constant).

The queued packets are stored in router buffer, of finite size. Under the common "drop tail" queueing discipline, any packets that arrive when the buffer is already full, are discarded.

Routing

Packets transmitted by a source node traverses the network, under the control of the routing system. The combination of network topology and routing is conveniently described as a routing matrix (see Sec. 2.3). This is a matrix with one row for each link in the network, and one column for each data flow. A data flow is characterized by its source node, its destination, and its sending rate. In this model, the destination node is passive, so we will usually say "source" then we really mean a flow and a pair of nodes. The sending rate of source s is denoted $r_s(t)$.

With this abstract notion of data sources, the routing matrix is defined as

$$A_{\ell s} = \begin{cases} 1 & \text{if the flow of source } s \text{ traverses link } \ell \\ 0 & \text{otherwise} \end{cases}$$

Links that are known not to be bottlenecks can be omitted.

Example 3.1 (Small network) Consider the network in Fig. 3.2. Assume that source S_k is sending data to node D_k, using shortest path routing, and that the

links marked L_1, L_2, and L_3 are the potential bottlenecks. The routing matrix for these three data flows and three links is

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

A corresponding graph is shown in Fig. 3.3. In this view, the intermediate nodes in the graph now represents individual links, and their corresponding queues. This is different from the view in Fig. 3.2, where intermediate nodes represents routers.

The routing matrix lets us describe *aggregation* easily. The arrival rates for the three links are given by

$$\begin{aligned} y_1 &= r_1 \\ y_2 &= r_1 + r_3 \\ y_3 &= r_2 + r_3 \end{aligned}$$

which can be written more concisely in matrix form,

$$y = Ar \tag{3.1}$$

□

Eq. (3.1) expresses the aggregation for any network and its corresponding routing matrix. It says that the arrival rate to link ℓ is the sum of the sending rates of all sources that uses that link. There are two important simplifications in this model.

Links are unordered Column s of the routing matrix represents the *unordered* set of links traversed by the packets transmitted by source s. But in a real network, order matters. E.g., consider a flow with sending rate 2 Mbit/s, that traverses two links of capacity 1 Mbit/s. Then the arrival rate to the first bottleneck is 2 Mbit/s, but this throttles the rate, so that the arrival rate to the second bottleneck is only 1 Mbit/s (or less, if the first bottleneck is shared with other traffic).

The throttling occurs only when the arrival rate to a link exceeds capacity, and then rates are throttled by a factor c_ℓ/y_ℓ. It follows that the discrepancy is small when the system is close to an equilibrium, where the arrival rate to each link is either below capacity, or close to the capacity.

No time delays The model does not include any delay between r_s and y_ℓ. Let $\tau_{\ell s}(t)$ denote the delay from the time a packet is transmitted by source s, until it arrives at link ℓ. Then $\tau_{\ell s}$ is a sum of propagation delays and the queueing delays at earlier links, evaluated at the appropriate points in the past. Then the aggregation can be written as

$$y_\ell(t) = \sum_{s;A_{\ell s}=1} r_s(t - \tau_{\ell s})$$

Taking the time-varying queueing delay into account leads to severe difficulties. For local analysis, it is reasonable to approximate $\tau_{\ell s}$ as the sum of propagation delays and stationary queueing delays. With this approximation of constant delays, the aggregation is linear and time-invariant. It can be expressed as a modified routing matrix, where all ones are replaced by time-shift operators. This makes it possible, although still quite difficult, to take delays into account for local stability analysis over arbitrary topologies.

Cross-traffic

When analyzing congestion control methods, it is useful to divide the system traffic into two types: Traffic from flows that use congestion control, and other traffic. We let y_ℓ denote the aggregation of the congestion controlled flows only, and let $x(t)$ denote the arrival rate of cross-traffic to each link. This gives the modified equation for the queueing,

$$\dot{q}_\ell(t) = \begin{cases} y_\ell(t) + x_\ell(t) - c_\ell(t) & q_\ell(t) > 0 \\ \max(0, y_\ell(t) + x_\ell(t) - c_\ell) & q_\ell(t) = 0 \end{cases}$$

The motivation for this division is that the rate of the cross-traffic is independent of the queue sizes in the network. E.g., fixed rate voice calls, or streaming video. In the analysis, we will usually assume that x is constant, and define the parameter $\gamma_\ell = 1 - x_\ell/c_\ell$, which is the proportion of capacity that is available for the congestion control flows.

One of the key observations is that the dynamical behavior of the system depends on γ. A 1 Mbit/s link with no cross-traffic ($\gamma = 1$) is very different from a 5 Mbit/s link with 80% cross-traffic ($\gamma = 0.2$), even though the available capacity is the same in both cases, $\gamma c = 1$ Mbit/s.

To illustrate this point, consider Figs. 3.4– 3.7 which show `ns2` simulations of TCP Vegas. We use a single bottleneck, with available capacity γc of 1 Mbit/s, and different values for γ. For all γ, TCP Vegas makes the queueing delay converge to the same value, 30 ms, corresponding to the transmission time for three packets at a rate of 1 Mbit/s. But the dynamics depend on γ: Small gamma gives a longer time constant, while high γ gives a short time constant and a large overshoot.

3.3 From window size to sending rate

In fluid flow modeling of queueing networks, data rates are the central quantities of interest. The rate of change of a queue is the difference between the arrival rate and the link capacity. On the other hand, in the TCP family of congestion control, the main control variable is the congestion window. The sending rate is determined indirectly by the window size, under the control of the ACK-clock. This can be described as a cascaded control system, illustrated in Fig. 3.1.

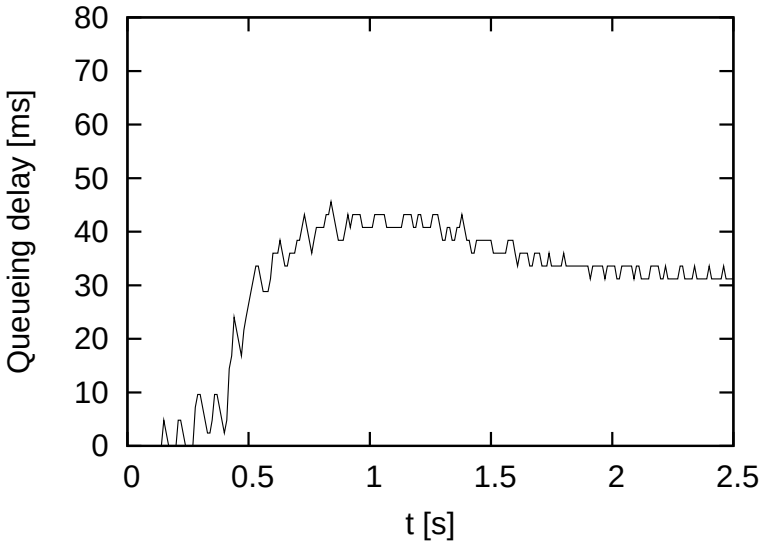

Figure 3.4: Queue response for TCP Vegas, for $\gamma = 0.2$.

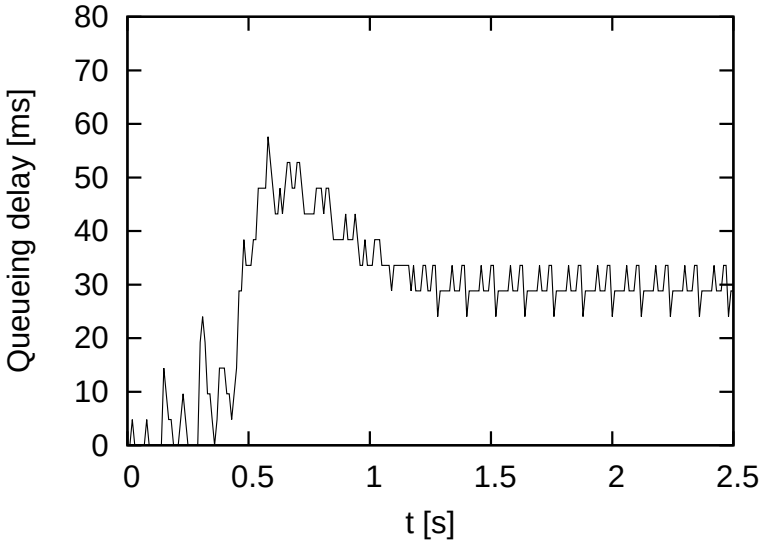

Figure 3.5: Queue response for TCP Vegas, for $\gamma = 0.4$.

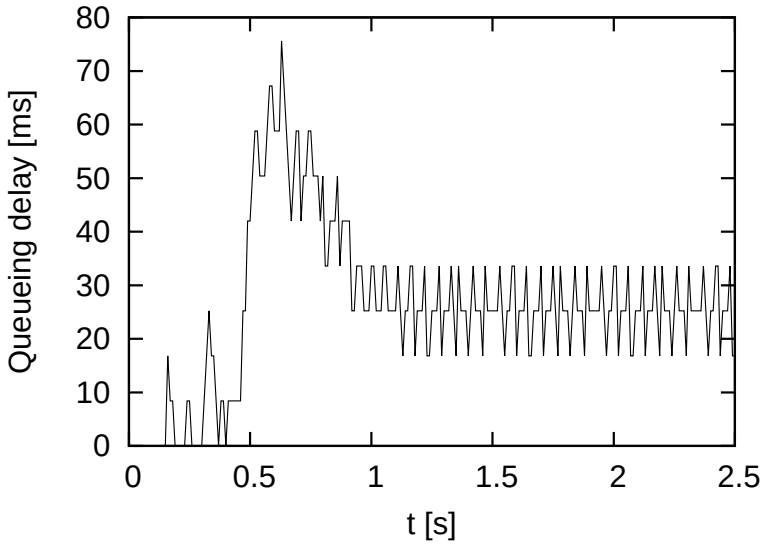

Figure 3.6: Queue response for TCP Vegas, for $\gamma = 0.7$.

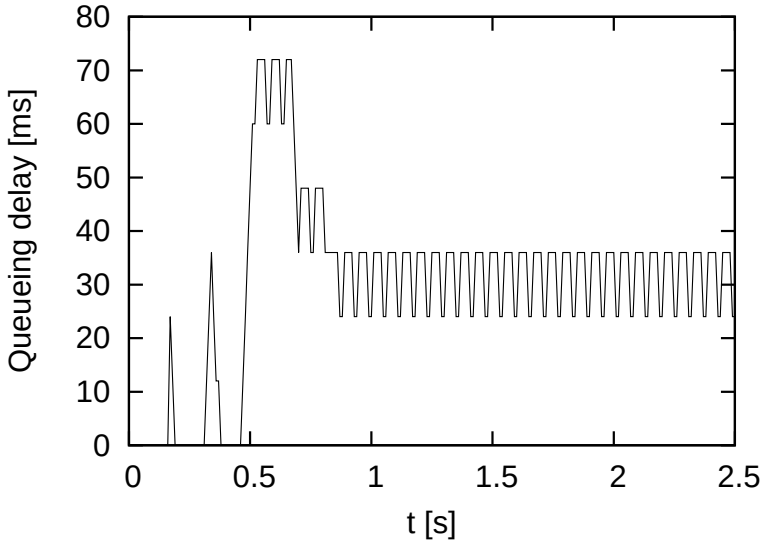

Figure 3.7: Queue response for TCP Vegas, for $\gamma = 1.0$.

Let us start with the relation between window size and the resulting sending rate. For equilibrium or average quantities, that relation is simple:

$$\text{sending rate} = \frac{\text{window size}}{\text{roundtrip time}}$$

but the dynamics are more complex than that. Let us first consider what is meant by "window size", since there is one important subtlety here. On one hand, we have the actual amount of in-flight data; packets that have been transmitted but for which no ACK has been received yet. This value will be denoted $w(t)$. On the other hand, we have the congestion window in TCP, `cwnd`, which is determined by the window update mechanism.

There may be a significant discrepancy between w and `cwnd`. For example, assume that as some point, both `cwnd` and w are ten packets, and that `cwnd` is reduced instantaneously from ten packets to five. To make w, the actual number of packets in flight, track this change instantaneously would require annihilation of five packets that are on their way somewhere else in the network. Obviously, that will not happen. Instead, the sending rate is reduced to zero, and w will drop by one packet at a time as ACKs arrive without any new packets being transmitted. Mathematically, this discrepancy is caused by the non-negativity constraint on the sending rate. Also, any upper bound on the sending rate will cause a discrepancy when `cwnd` is increased.

When `cwnd` is used as an input signal to the ACK-clock inner-loop, the actual amount of in-flight data, w will track `cwnd` as closely as possible given the constraints on the sending rate. When the rate of change of `cwnd` is small, the discrepancy will be small. When using the model, the distinction between w and `cwnd` will usually be ignored.

Forward and backward delay

Consider a single sender, maintaining an amount of $w(t)$ in-flight data at time t, and a sending rate $r(t)$. The data is transmitted over a path with a single bottleneck link of capacity c, and a roundtrip propagation delay τ. The queue that serves the bottleneck link is called the *bottleneck queue*.

The propagation delay can be divided into forward delay τ_f (the time it takes for a transmitted packet to arrive to the bottleneck queue) and backward delay τ_b (the time it takes for a packet transmitted on the bottleneck link to arrive to the receiver, and for the corresponding ACK to travel back to the sender).

To see the influence of the forward delay, consider two single-bottleneck single-flow networks, with the same roundtrip propagation delay τ. In the first network, the forward delay is $\tau_f > 0$. The sender uses a window size of $w_1(t)$ and a sending rate of $r(t)$. In the second network, the sender is located just before the bottleneck so that the forward delay is zero. The sender in the second network uses a sending rate of $r_2(t) = r(t - \tau_f)$. Comparing these two networks, the arrival rate to the

bottleneck, in both cases, is $r(t-\tau_f)$. It follows that the queue size $q(t)$, as well as the sending rate on the bottleneck link, is identical for both networks.

Then the rates of received ACKs are also the same in both networks, except for a time shift due to the difference in backward delay, which is τ_f. Since the window size is the difference between transmitted packets and ACKed packets, it follows that the window size in the second network is $w_2(t) = w_1(t-\tau_f)$. This argument extends to any flow that traverses a single bottleneck, within a general network topology with several flows and several bottlenecks.

Proposition 3.2 (Informal) *Assume that some flow traverses a single bottleneck, with a forward delay τ_f and window size $w(t)$.*

(i) The queue size $q(t)$ at the bottleneck is unchanged if the flow is replaced by a flow with the same roundtrip propagation delay, zero forward delay, and a window size of $w(t-\tau_f)$

(ii) Except for this delay in the input signal, the queue dynamics depends only on the total propagation delay of the flow, not on its subdivision into forward and backward delay.

Integrator model

In the literature, there are two common ways of expressing the sending rate of this system. The first approach applies the relation

$$\text{sending rate} = \frac{\text{window size}}{\text{roundtrip time}},$$

to the instantaneous rate,

$$r(t) = \frac{w(t)}{\tau + q(t-\tau_b)/c}$$

The time delay in the denominator is motivated by causality. Information about the queue size travels back to the sender with the packets and ACKs, which are subject to the propagation delay τ_b. It also ensures that feedback delay in queueing dynamics will correspond to the total propagation delay, not just the forward delay.

Together with the integrating queue dynamics, and cross-traffic $x(t)$, one gets the *integrator link model*

$$\dot{q}(t) = \frac{w(t-\tau_f)}{\tau + q(t-\tau)/c} + x(t) - c \tag{3.2}$$

A similar model, generalized to an arbitrary network topology, is described in [80]. The topology is specified by the routing matrix A, equilibrium forward delays $\tau_{\ell s}^{f*}$, and propagation round trip delays τ_s, as

$$\dot{q}_\ell(t) = \sum_s A_{\ell s} \frac{w_s(t-\tau_{\ell s}^f)}{\tau_s + \sum_k A_{ks} q_k(t-\tau_{\ell s}^{f*})/c_k} - c_\ell$$

In the case of a single flow, single source, this reduces to

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

In this model, τ_f^* includes forward propagation delay and equilibrium queueing delay, but not backward propagation delay, hence this model violates Prop. 3.2.

In [56], the queue dynamics is written as

$$\dot{q}(t) = \frac{w(t)}{\tau + q(t)/c} n(t) - c$$

where $n(t)$ is the number of TCP flows sharing the bottleneck. For $n = 1$, this is the same integrator model (3.2) with zero cross-traffic, except for the missing delay.

The models used in [56, 80] boil down to the relation $r(t) = w(t)/(\tau + q(t)/c)$, connecting sending rate and window size, without any delay. It should be noted that unlike the present chapter, this part of the model is not the main focus in these papers; the main modeling complexity is in the window update law and the AQM scheme.

Static link model

The second approach is based on the observation that the window size is the amount of data that is in transit somewhere in the network. Let us first consider the case with no cross-traffic, $x(t) = 0$. Of the data in transit, all but at most $c\tau$ must reside in the queue at the bottleneck link. This leads to the *static link model*

$$q(t) = w(t) - c\tau$$

This corresponds to a sending rate of

$$r(t) = c + \dot{w}(t)$$

This expression for the rate can be understood as follows: Packets are transmitted onto the bottleneck link with rate c. Then the rate of received ACKs will also be c, independent of the window size [114]. So one can read the equation as

$$r(t) = \text{rate of received ACKs} + \dot{w}(t)$$

To extend the static model to the case with cross-traffic, we only consider the case of constant cross-traffic. Let $\gamma = 1 - x/c$, i.e., the proportion of the capacity that is available. Looking for a static relation between w and q, the corresponding equations are

$$q(t) = \frac{w(t)}{\gamma} - c\tau$$

$$r(t) = \gamma c + \frac{\dot{w}(t)}{\gamma}$$

This is the only static relation between w and q which gives the correct equilibrium, $r = \gamma c\tau$ and $w = \gamma(c\tau + q)$.

3.4 New joint link model

The integrator model and the static model are obviously contradictory. It turns out that the integrator model is accurate when the bottleneck link is dominated by cross-traffic, while the static model is accurate when the cross-traffic uses only a small proportion of the capacity. We will now derive a model that is only slightly more complex, and which is accurate for any level of cross-traffic. We will see that in the low cross-traffic limit, we recover the static model, while in the high cross-traffic limit, we recover the integrator model.

Assume we have a single flow traversing a single bottleneck. Consider the transmission at some time t_0, and put $t_1 = t_0 + \tau + q(t_0 + \tau_\mathrm{f})/c$. Then the ACK for a packet sent at time $t = t_0$ will return to the sender at time $t = t_1$. Furthermore, at time t_1, the ACKs for all packets sent before t_0 have returned, which means that the data in flight at time t_1 is precisely the data sent during the interval $t_0 < t \leq t_1$. Mathematically,

$$w(t_1) = \int_{t_0}^{t_1} r(s)\,\mathrm{d}s$$

Rewrite the left hand side as $w(t_1) = w(t_0) + \int_{t_0}^{t_1} \dot{w}(t)\,\mathrm{d}t$, and move the integral to the right-hand side. Then, by the mean value theorem, there exists a point ξ in the interval $t_0 \leq \xi \leq t_1$ such that

$$w(t_0) = \int_{t_0}^{t_1} \big(r(s) - \dot{w}(s)\big)\,\mathrm{d}s = (t_1 - t_0)\,\big(r(\xi) - \dot{w}(\xi)\big)$$

Solving for r, we get

$$r(\xi) = \frac{w(t_0)}{\tau + q(t_0 + \tau_\mathrm{f})/c} + \dot{w}(\xi)$$

The time delay between q and r is $\xi - (t_0 + \tau_\mathrm{f})$. Like ξ, this delay is unknown and most likely dependent both on t_0 and on the initial conditions. To make the model practical, we need to approximate the delay with a constant, $\xi = t_0 + \Delta$. Since $\xi \leq t_1$, we require that $\Delta \leq \tau + q(t_0 + \tau_\mathrm{f})/c$. For this bound to be valid for all t_0 and all initial conditions, including $q(t_0 + \tau_\mathrm{f}) = 0$, we must have $\Delta \leq \tau$. On the other hand, causality requires $\xi - (t_0 + \tau_\mathrm{f}) \geq \tau_\mathrm{b}$, which implies that $\Delta \geq \tau$.

This leads us to the approximation $\xi = t_0 + \tau$. The resulting equation is

$$r(t) \approx \frac{w(t - \tau)}{\tau + q(t - \tau_\mathrm{b})/c} + \dot{w}(t)$$

The time delays in the real system include queueing delays, and are hence state-dependent. Any choice of constant delays in the model is arbitrary to some degree. The choice of fixed delays in the model is a compromise between simplicity and accuracy, with causality requirements taken into account.

The delay in the denominator is clearly motivated by causality; τ_b is the signalling delay between the queue and the sender. The delay in the numerator may

at first seem unintuitive; why should there be a signalling delay from the window size, a direct input to the sender's transmission control, to the resulting rate? It can be understood if we interpret the first term as the rate of received ACKs. This rate depends on the window size at the time the corresponding data was sent, and the queue size at the time the data arrived to the queue. The second term represents additional data that is sent or omitted immediately when w is changed.

If we substitute the rate expression into the queue dynamics, $\dot{q}(t) = r(t - \tau_f) + x(t) - c$, we get

$$\dot{q}(t) = \frac{w(t - \tau - \tau_f)}{\tau + q(t - \tau)/c} + \dot{w}(t - \tau_f) + x(t) - c$$

We call this model the *joint link model*.

Linearized dynamics

To examine behavior close to the equilibrium, assume that the cross-traffic is constant, and define $\gamma = 1 - x/c$. This is the proportion of the capacity that is available, i.e., not used by cross-traffic. Without loss of generality, we can also assume that $\tau_f = 0$. We examine the response to small changes in the window size close to an equilibrium w^* and q^*. If $w^* \leq \gamma c\tau$, then the equilibrium is $q^* = 0$, with an active non-negativity constraint in the equation for $\dot{q}(t)$. To get a non-empty equilibrium queue, we must require that $w^* > \gamma c\tau$. In this case, $q^* = w^*/\gamma - c\tau$. Also let $\tau^* = \tau + q^*/c$ denote the total RTT in equilibrium.

Put $w(t) = w^* + \widetilde{w}(t)$ and $q(t) = q^* + \widetilde{q}(t)$. We then get

$$\dot{\widetilde{q}}(t) = \frac{\widetilde{w}(t - \tau) - \gamma\widetilde{q}(t - \tau)}{\tau^* + \widetilde{q}(t - \tau)/c} + \dot{\widetilde{w}}(t)$$

Linearization and transform into the Laplace domain gives the following transfer function from $\widetilde{w}$ to $\widetilde{q}$,

$$G(s) = \frac{e^{-s\tau} + \tau^* s}{\gamma e^{-s\tau} + \tau^* s}$$

Remarks:

- The static gain is $G(s) = 1/\gamma$. I.e., the effect a window change has on the queue is larger the more cross-traffic there is. This can also be seen directly from the equilibrium equation, $q^* = w^*/\gamma - c\tau$.
- In the limit $\gamma \to 1$, we get $G(s) \to 1$, which is the static model.
- The integrator model has the same equilibrium equation, and the corresponding linearization gives the transfer function

$$G_{\text{integrator}}(s) = \frac{1}{\gamma e^{-\tau s} + \tau^* s}$$

This implies that the poles, stability properties, and the static gain, are the same for the joint model and the integrator model. The difference is that the joint model has additional zeros.

- The system poles are the roots of $\gamma e^{-s\tau} + \tau^* s$. These can be expressed using Lambert's function (see Appendix B), as $s_k = W_k(-\gamma\tau/\tau^*)/\tau$, where k enumerates the different branches of the function. Stability, and bounds for the poles, will be considered in the next chapter.

- If γ is small, $\gamma < 1/e$ (or more generally, if $\gamma\tau/\tau^* < 1/e$), then there are two real poles, on the negative real axis. The real pole closest to the origin is $s_0 = W_0(-\gamma\tau/\tau^*)/\tau$. Since $W_0(x) = x + O(x^2)$, the real pole closest to the origin is located at

$$s_0 = \frac{1}{\tau} W_0\left(-\gamma\frac{\tau}{\tau^*}\right) = -\frac{\gamma}{\tau^*} + O(\gamma^2)$$

  Hence, for small γ, the system time constant is roughly τ^*/γ, or possibly larger.

3.5 Validation

To validate the new model, we compare it to `ns2` simulations of TCP, using a configuration where only the ACK-clock inner-loop is active, and the additive increase and multiplicative decrease mechanisms are disabled.

The simulation scenario used a single-bottleneck topology, with 1 Mbit/s available capacity (i.e., $\gamma c = 1$ Mbit/s), and propagation roundtrip time of 50 ms. The cross-traffic is constant bit rate UDP, and both the TCP flow and the cross-traffic use a packet size of 1500 bytes. The initial window size was chosen to get a queue size of roughly 20 packets. Figure 3.8 and 3.9 show the response to a positive and negative step in the window size. We see that for smaller γ, i.e., more cross-traffic, the resulting change in the queue size is larger, and the time to converge to the new value is longer.

Now compare these curves to the queue size predicted by the the three models. In each of Fig 3.10–3.13, the curves are plotted together for one value of γ. We see that the integrator model is accurate for high cross-traffic (small γ), while the static model is accurate for low cross-traffic (large γ), the joint model is fairly close to the `ns2` simulation regardless of the intensity of cross-traffic.

Finally, let us revisit the issue of where in the model we should put the time

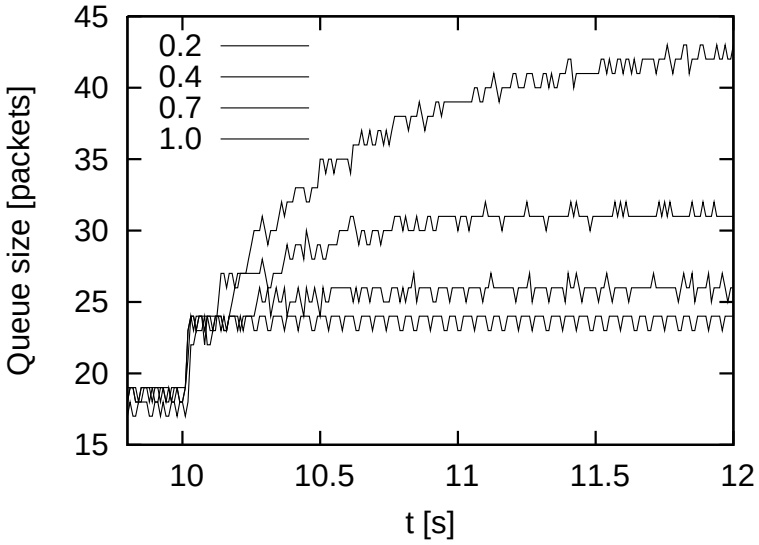

Figure 3.8: Queue response to a positive step change in the window size. The window size is increased by five packets at time $t = 10$. From top to bottom, the curves correspond to increasing γ.

delays. Consider the three candidate models:

$$\dot{q}(t) = \frac{w(t)}{\tau + q(t)/c} + \dot{w}(t) - \gamma c \qquad \text{No signalling delay} \qquad (3.3)$$

$$\dot{q}(t) = \frac{w(t)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c \qquad \text{Only } q \text{ subject to delay} \qquad (3.4)$$

$$\dot{q}(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c \qquad \text{Both } q \text{ and } w \text{ subject to delay} \qquad (3.5)$$

In Fig. 3.14, we see the same step response experiment as in Fig 3.11, together with the predictions from these three model variants. The model with time delay for q only, Eq. (3.4), gives worse accuracy then the model with no time delays at all, Eq. (3.3). So it seems essential to use the same delay for both numerator and denominator in the expression $w/(\tau + q/c)$.

Signalling delays in a system are of fundamental importance both for stability and for the achievable performance. Since we want results that hold also when the propagation delay is large, we will prefer to use the model with time delays, Eq. (3.5). When we need a simpler model, e.g., in order to get closed form solutions, we will resort to Eq. (3.3), the model without signalling delays, which is a fairly good approximation.

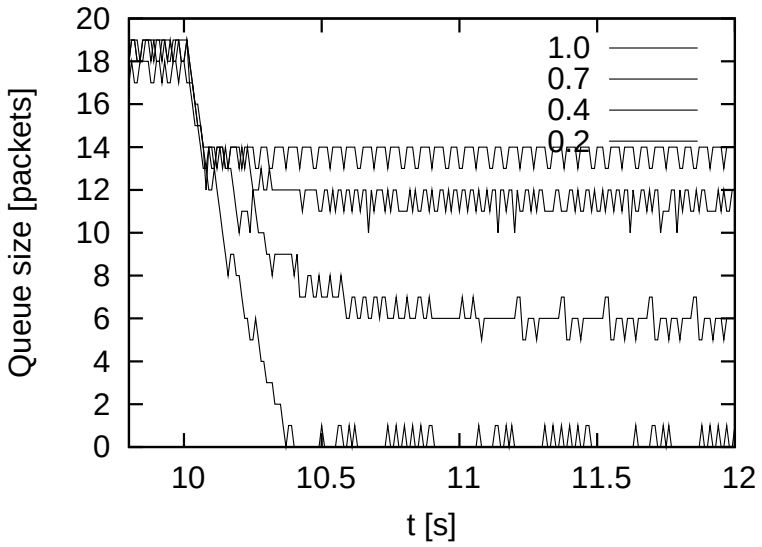

Figure 3.9: Queue response to a negative step change in the window size. The window size is decreased by five packets at time $t = 10$. From top to bottom, the curves correspond to decreasing γ.

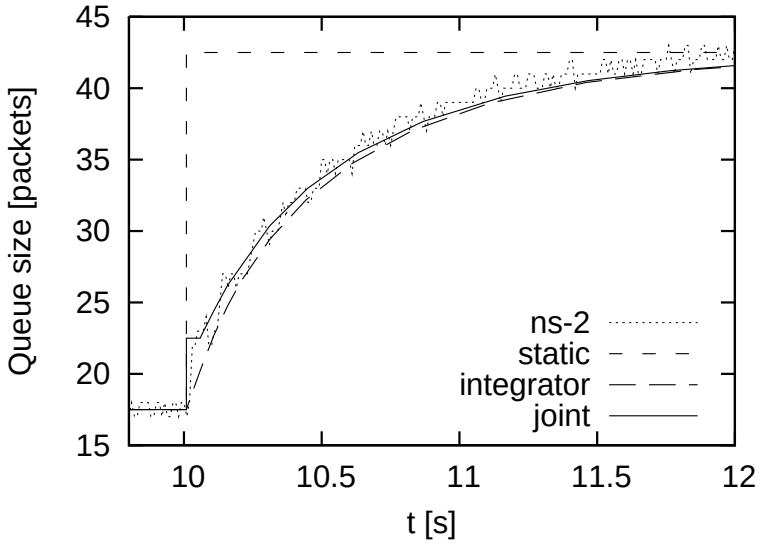

Figure 3.10: Queue response from `ns2` compared to the prediction of three queue models, for $\gamma = 0.2$.

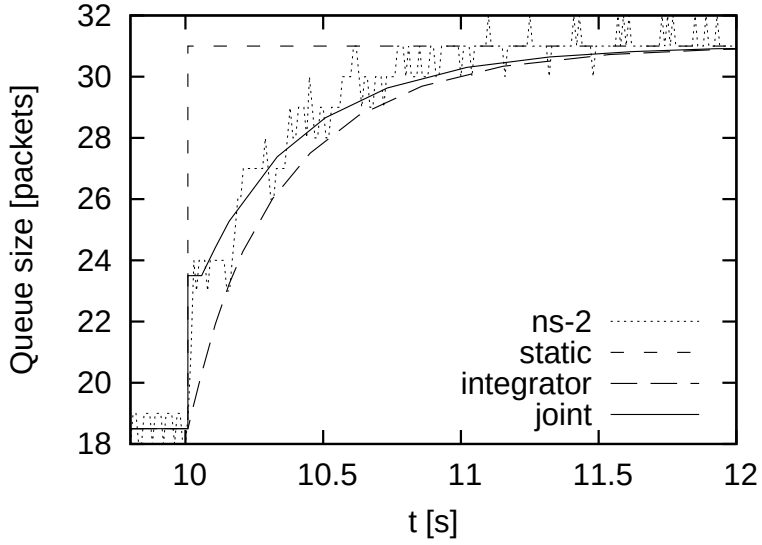

Figure 3.11: Queue response from `ns2` compared to the prediction of three queue models, for $\gamma = 0.4$.

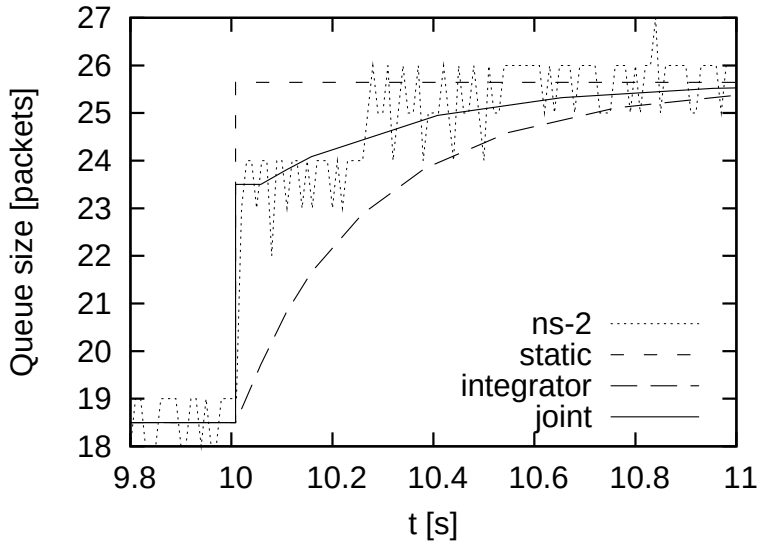

Figure 3.12: Queue response from `ns2` compared to the prediction of three queue models, for $\gamma = 0.7$.

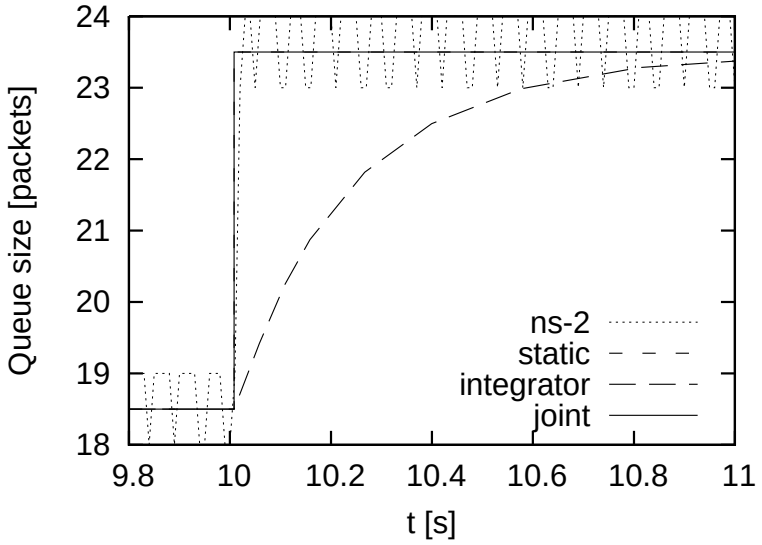

Figure 3.13: Queue response from `ns2` compared to the prediction of three queue models, for $\gamma = 1.0$. For this scenario, with no cross-traffic, the curves for the static model and the joint model are the same.

3.6 Multiple flows

In the joint link model for a single flow, we had one flow using window-based control, sharing the network with cross-traffic. The cross-traffic arrival rate was assumed to be independent of the queue size. In this section, we extend the model to the case of single link shared by n flows using window-based control, all with sending rates depending on the queue size, as well as some cross-traffic.

Figure 3.15 illustrates the topology, for $n = 5$. For flow k, we have a window size $w_k(t)$, sending rate $r_k(t)$, and a roundtrip propagation delay τ_k. Without loss of generality, we can assume that the forward delay is zero. The equation

$$w_k(t_0 + \tau_k + q(t_0)/c) = \int_{t_0}^{t_0+\tau_k+q(t_0)/c} r_k(t)\,\mathrm{d}t$$

implies

$$r_k(\xi) = \frac{w_k(t_0)}{\tau_k + q(t_0)/c} + \dot{w}_k(\xi)$$

for some ξ with $t_0 \le \xi \le t_0 + \tau_k + q(t_0)/c$. Like in the single flow case, the approximation $\xi = t_0 + \tau_k$ is a reasonable compromise. The dynamics of the bottleneck

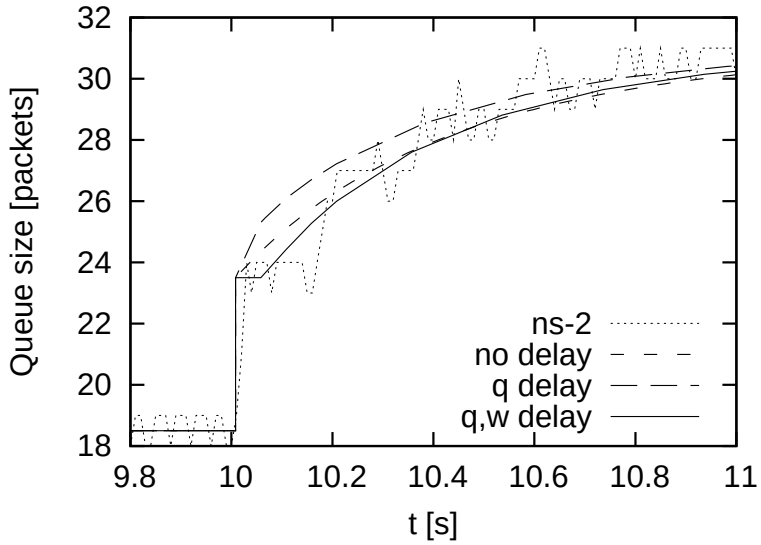

Figure 3.14: Queue response for three models with three different choices of signalling delays, compared to `ns2` simulation. The window size is increased by five packets at time $t = 10$, and $\gamma = 0.4$.

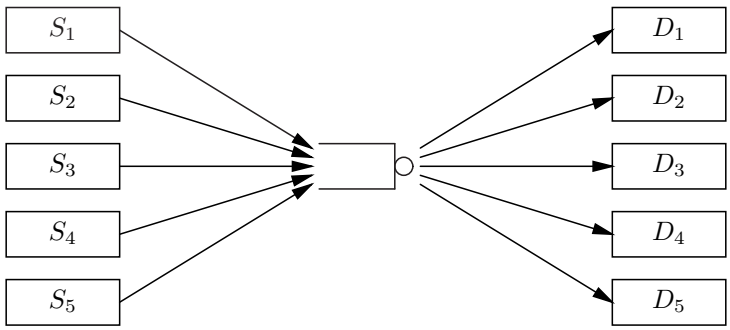

Figure 3.15: Topology with a single bottleneck shared by several flows.

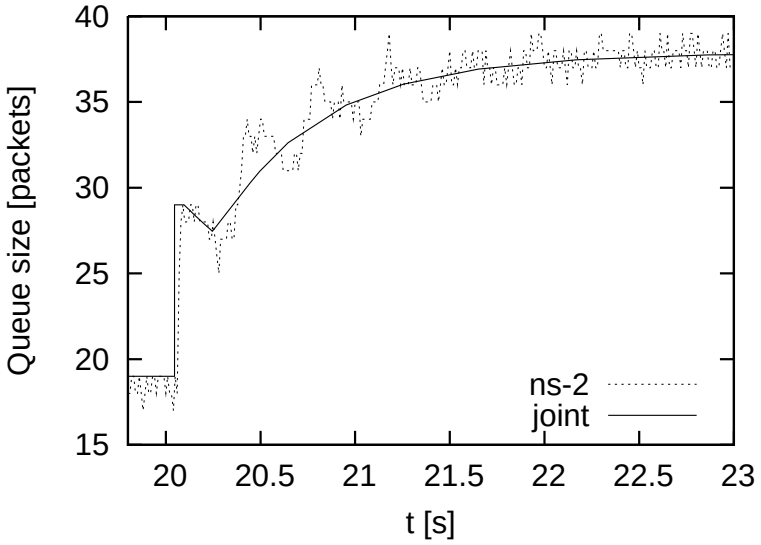

Figure 3.16: Behavior with two flows different propagation delays, 50 ms and 200 ms. For $t < 20$, the queue is in equilibrium, each flow has a sending rate of 500 kbit/s, and there is also 1.5 Mbit/s of constant rate cross-traffic ($\gamma = 0.4$). At time $t = 20$, the window size of the flow with the longer propagation delay is increased by ten packets.

queue is then given by

$$\dot{q}(t) = \sum_k r_k(t) + x(t) - c = \sum_k \dot{w}_k(t) + \sum_k \frac{w_k(t-\tau_k)}{\tau_k + q(t-\tau_k)/c} + x(t) - c$$

Flows with the same roundtrip propagation delay behave as a single aggregated flow, where the rate and the window size of the aggregate are the sums of the rate and window sizes of the individual flows. With heterogeneous delays, the dynamics get more complex. In Fig. 3.16, we see the queue response to a change in window size, for two flows with a factor four difference in RTT. The model is quite close to the `ns2` simulation, although there are some unmodelled oscillations. The the oscillations seem related to packet bursts, the initial burst when the window is increased, and repeated bursts when the ACKs for the previous burst return to the sender.

When the difference in RTT becomes more extreme, the oscillations are more pronounced, as seen in Fig. 3.17, with a factor 16 difference in RTT. Introduction of traffic shaping at the senders, to reduce bursts, may reduce the oscillations and bring the dynamics closer to that described by the model. The effect of traffic shaping, and the nature of these oscillations, needs further investigation.

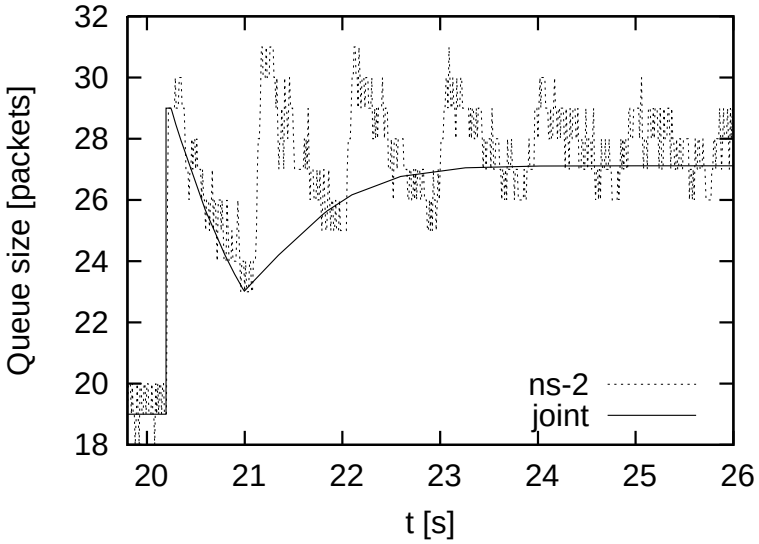

Figure 3.17: Behavior with two flows with propagation delays that differ by an order of magnitude, 50 ms and 800 ms. For $t < 20$, the queue is in equilibrium, each flow has a sending rate of 500 kbit/s, and there is also 1.5 Mbit/s of constant rate cross-traffic ($\gamma = 0.4$). At time $t = 20$, the window size of the flow with the longer propagation delay is increased by ten packets.

3.7 Equilibrium for a general network

In this section, consider a general network consisting of S sources, each sending data across the network using a fixed window size $w_s > 0$ and a propagation delay τ_s^0, and L links, each with a fixed capacity $c_\ell > 0$. The interconnection of sources and links is given by the routing matrix A (Sec. 2.3), defined by

$$A_{\ell s} = \begin{cases} 1 & \text{if source } s \text{ uses link } \ell \\ 0 & \text{otherwise} \end{cases}$$

Let $d_\ell(t) = q_\ell(t)/c_\ell$ denote the queueing delay at link ℓ, and let $r_s(t)$ be the sending rate of source s. The aggregated arrival rates at the links are $y = Ar$, and the aggregates queueing delays seen by the sources are $\tau = A^T d$. Since we will examine equilibrium properties only, we can ignore signalling delays. Then the sending rate for each source is

$$r_s(t) = \frac{w_s}{\tau_s^0 + \tau_s(t)}$$

and the queue dynamics at each link is

$$\dot{d}_\ell(t) = \begin{cases} \frac{y_\ell(t)}{c_\ell} - 1 & d_\ell(t) > 0 \\ \max\left(0, \frac{y_\ell(t)}{c_\ell} - 1\right) & d_\ell(t) = 0 \end{cases} \tag{3.6}$$

The equilibrium condition $\dot{d}_\ell = 0$ implies that either $d_\ell = 0$ and $y_\ell \leq c_\ell$, or $y_\ell = c_\ell$. Equivalently, in vector notation, $y \leq c$ and $d^T(c - y) = 0$. To sum up, for each stationary vector d, with corresponding sending rates r, the equilibrium conditions can be written as

$r \geq 0$	$d \geq 0$	non-negativity
$y \leq c$		capacity constraint
$y = Ar$	$\tau = A^T d$	aggregation
$r_s = \dfrac{w_s}{\tau_s^0 + \tau_s}$		rate equation
$0 = d^T(c - y)$		complementary slackness

This looks very similar to the KKT conditions for a network utility maximization problem (Sec. 2.3). If we solve the rate equation for τ_s, we get

$$\tau_s = \frac{w_s}{r_s} - \tau_s^0$$

Define

$$U_s(r_s) = w_s \log r_s - \tau_s^0 r_s$$

This is a strictly concave function. Unlike the utility functions commonly used in network utility maximization, U_s is not increasing for all r_s, it has a maximum at $r_s = w_s/\tau_s^0$. We thus get equivalence between equilibria of (3.6) and a convex optimization problem.

Theorem 3.3 *For any given windows sizes $w > 0$, and capacities $c > 0$, and any routing matrix A with full row rank, the dynamical system* (3.6) *has a uniquely determined equilibrium d^*.*

Proof: Consider the utility maximization problem

$$\begin{aligned} \max_r \quad & \sum_s (w_s \log r_s - \tau_s^0 r_s) \\ \text{s.t.} \quad & Ar \leq c \\ & r \geq 0 \end{aligned}$$

The constraints define a convex region, with nonempty interior (since $c > 0$), and the objective function is strictly concave (since $w > 0$). Hence, this is a convex

optimization problem, with a unique maximizer r^*, characterized by the KKT conditions. We have seen that the KKT conditions are equivalent to the equilibrium equations for (3.6). For any pair of vectors r^* and d^* that satisfy the KKT conditions, d^* is an equilibrium. This establishes existence of an equilibrium.

For uniqueness, assume that d and $\widetilde{d}$ are two equilibria. Together with the corresponding rates, we get two solutions to the KKT equations. By uniqueness of the maximizer, r^* is uniquely determined, i.e., the two equilibria correspond to the same rates. By the rate equation, the aggregated delay τ is uniquely determined by r^*. By the aggregation equation, it follows that $A^T(d - \widetilde{d}) = 0$. This expression is a linear combination of the rows of A. Since A has full row rank, the rows of A are linearly independent, and it follows that $d = \widetilde{d}$. Hence the equilibrium d^* is uniquely determined. □

This theorem is proved in a different manner in [89].

Example 3.4 (Equilibrium queue sizes) Recall the topology in Fig. 3.3. There are three flows and three links, each with one router queue. The routing matrix for this network is

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Let us assume that all links have the same capacity 10 Mbit/s, and propagation delay 20 ms, for both data packets and ACKs. ACKs are assumed to traverse the same number of routers and links, but with uncongested queues. Also assume that the three window sizes are 64 Kbyte $= 2^{19}$ bits. Then the parameters of this system are

$$c = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix} 10^6 \qquad \tau^0 = \begin{pmatrix} 80 \\ 40 \\ 80 \end{pmatrix} 10^{-3} \qquad w = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} 2^{19}$$

Then the equilibrium of the queueing dynamics, Eq. (3.6), is

$$r = \begin{pmatrix} 6.06 \\ 6.06 \\ 3.94 \end{pmatrix} 10^6 \qquad y = \begin{pmatrix} 6.06 \\ 10 \\ 10 \end{pmatrix} 10^6 \qquad d = \begin{pmatrix} 0 \\ 6.5 \\ 46.5 \end{pmatrix} 10^{-3}$$

We see that only the links L_2 and L_3 are bottlenecks. If we reduce the capacity of L_1 to 5 Mbit/s, we get a different solution, where the bottlenecks are L_1 and L_3.

$$r = \begin{pmatrix} 5 \\ 5.92 \\ 4.08 \end{pmatrix} 10^6 \qquad y = \begin{pmatrix} 5 \\ 9.08 \\ 10 \end{pmatrix} 10^6 \qquad d = \begin{pmatrix} 25.9 \\ 0 \\ 48.5 \end{pmatrix} 10^{-3}$$

□

3.8 Summary

In this chapter, we have proposed and validated modeling of the relationship between the window size and the sending rate as

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

When combined with the integrator queueing dynamics in a single source, single link topology, the resulting queueing dynamics is

$$\dot{q}(t) = \frac{w(t-\tau-\tau_{\mathrm{f}})}{\tau + q(t-\tau)/c} + \dot{w}(t-\tau_{\mathrm{f}}) + x(t) - c$$

This model unifies the integrator model, $r(t) = w(t)/(\tau + q(t-\tau)/c)$, and the static model, $q(t) = w(t) - c\tau$.

By comparing all three models to `ns2` simulations, we have seen that the integrator model is accurate for a high level of cross-traffic, while the static model is accurate for a low level of cross-traffic. The new joint model describes the queueing dynamics accurately for any level of cross-traffic. In Chapter 5, we will also use a simplified joint model, without signalling delays:

$$r(t) = \frac{w(t)}{\tau + q(t)/c} + \dot{w}(t) \tag{3.7}$$

For constant cross-traffic, we use the parameter $\gamma = (c - x)/c$. This is the proportion of the capacity that is available, i.e., not used by cross-traffic. The static gain from window size to queue size is $1/\gamma$, hence increasing with the amount of cross-traffic. Since for small γ, the linearized system has a pole located at $-\gamma/\tau^* + O(\gamma^2)$, we expect that increased cross-traffic will also make the system slower. In this formula, τ^* denotes the total RTT in equilibrium.

In the next chapter, we will analyze the stability of this system, and give bounds for the convergence time constant.

Chapter 4

Stability analysis of window-based transmission

I admire your perverted ingenuity in inventing one definition after the other as barricades against the falsification of your pet ideas. Why don't you just define a polyhedron as a system of polygons for which the equation $V - E + F = 2$ holds?

Proofs and Refutations, Imre Lakatos

In this chapter, we argue that window-based congestion control is stable. More precisely, we consider a queueing network, where the sending rate of each traffic source is determined by the source's window size and the ACK-clock mechanism. The window sizes are the inputs to the system, and the queue sizes inside the network are the system outputs. We will also allow for constant-rate cross-traffic flows.

4.1 Overview

Proving global stability for the most general case, with arbitrary network topology and arbitrary delays, seems very difficult. Instead we consider the most important special cases. We have the following results:

- The case of a single bottleneck and a single flow is globally stable, for arbitrary delays. This is proved in Sec. 4.2. We also discuss the mathematical well-posedness of the problem, and the convergence time constant.
- The case of a single bottleneck and an arbitrary number of flows with arbitrary delays, is locally stable. This is proved in Sec. 4.3.
- With an arbitrary network topology, under the approximation of no signalling delays, the system is globally stable. This is proved in Sec. 4.4

These results indicate that neither delays, aggregation, or complex network topology, leads to any stability problems.

Since we study asymptotic stability with arbitrary but *constant* inputs, the direct term $\dot{w}$ in the joint link model vanishes, and the results are valid also for the integrator link model.

In Sec.4.5, we prove general theorems on the existence and uniqueness of solutions to differential equations with state constraints. The well-posedness result in Sec. 4.2 is a direct consequence of one of these theorems. In Sec. 4.6, we derive sufficient stability conditions for certain low-dimensional linear systems with delays. These results are used in this and later chapters, for proving local stability.

4.2 A single link and a single flow

Well-posedness

We start by looking closer at the differential equation for the queue evolution in the case of a single flow and a single bottleneck, and the existence and uniqueness of solutions. The queue dynamics are described by the *joint link model* from Chapter 3,

$$\dot{q}(t) = \begin{cases} f(t, q(t-\tau)) & q(t) > 0 \\ \max(0, f(t, q(t-\tau)) & q(t) = 0 \end{cases} \tag{4.1}$$
$$f(t,q) = \dot{w}(t) + \frac{w(t-\tau)}{\tau + q/c} + x(t) - c$$

The parameter $c > 0$ is the link capacity, and $\tau > 0$ is the roundtrip propagation delay, i.e., excluding the queueing delay $q(t)/c$. There are two input signals, the window size $w(t) \geq 0$, and the arrival rate of cross-traffic, $0 \leq x(t) < c$.

Theorem 4.1 (Well-posedness) *Assume the input signal $x(t)$ is continuous, and $w(t)$ is continuously differentiable. For any non-negative, right-hand continuous, bounded and measurable function ϕ_0 on $[-\tau, 0]$, Eq. (4.1), together with the initial condition $q(t) = \phi_0(t)$ for $t \in [-\tau, 0]$, has a uniquely determined solution $q(t)$ for all $t \geq 0$.*

Proof: Under these assumptions, f and ϕ_0 satisfy the hypothesis of Theorem 4.15, which is stated and proved in Sec. 4.5. □

Stability

Now restrict attention to the case of a constant input $w(t) = w$ and constant cross-traffic $x(t) = x$. Put $\gamma = (c - x)/c$; this is the proportion of the capacity that is not consumed by the cross-traffic. Then there is no longer an explicit time-dependence

in f, and the dynamics are described by

$$\dot{q}(t) = \begin{cases} f(q(t-\tau)) & q(t) > 0 \\ \max(0, f(q(t-\tau)) & q(t) = 0 \end{cases} \qquad (4.2)$$
$$f(q) = \frac{w}{\tau + q/c} - \gamma c$$

The equation has a unique equilibrium given by

$$q^* = \max(0, w/\gamma - c\tau)$$

The time delay is the source of any complex or unstable dynamics in this system; if $\tau = 0$, the system is trivially stable.

Theorem 4.2 (Global asymptotic stability) *For any non-negative, right-hand continuous, bounded and measurable function ϕ_0 on $[-\tau, 0]$, the solution to Eq. (4.2), with initial condition $q(t) = \phi_0(t)$ for $t \in [-\tau, 0]$, satisfies $q(t) \to q^*$ as $t \to \infty$.*

First consider the simplest case, $w \leq \gamma c\tau$. Then $q^* = 0$, and since $\dot{q}(t) \leq 0$ for all t, it follows that this equilibrium is globally asymptotically stable.

For the rest of this section, assume that $w > \gamma c\tau$, so that $q^* > 0$. Also put $\tau^* = \tau + q^*/c$. Then the equation for f can be rewritten once more, as

$$f(q) = \gamma \frac{q^* - q}{\tau + q/c}$$

Note that this is a strictly decreasing function of q. The stability proof depends on various bounds on f. The parameter γ plays limited rôle in this proof; in all inequalities, it will be eliminated using $\gamma \leq 1$.

We first show that the system is stable, under an additional assumption that trajectories are appropriately bounded.

Lemma 4.3 *Assume that $q(t)$ is a solution to Eq. (4.2), and that $q(t)$ satisfies the bounds $0 < q(t) \leq 2q^*$ for all $t \geq 0$. Then $q(t) \to q^*$ as $t \to \infty$.*

Proof: Since $q(t) > 0$ for all $t > 0$, the non-negativity condition is not active. We first make a change of variables

$$v(t) = \frac{q(t\tau) - q^*}{c\tau + q^*}$$

which transforms the equation into

$$\dot{v}(t) = -\kappa \frac{v(t-1)}{1 + v(t-1)}$$

where $\kappa = \gamma\tau/\tau^* < \gamma$. The bounds $0 < q(t) \leq 2q^*$ imply that $|r(t)| \leq 1 - \tau/\tau^* \leq 1 - \kappa$.

The differential equation can be decomposed as a linear system with a static non-linear feedback $\psi(x)$.

$$\begin{aligned}\dot{v}(t) &= -\kappa v(t-1) + \kappa u(t-1)\\ u(t) &= -\psi(v(t))\\ \psi(v) &= -\frac{v^2}{1+v}\end{aligned}$$

The transfer function for the linear system with $u(t)$ as input and $v(t)$ as output is

$$G(s) = \frac{\kappa e^{-s}}{s + \kappa e^{-s}}$$

which is stable (an application of the argument principle gives the necessary and sufficient stability condition $0 \le \kappa < \pi/2$, which is satisfied).

For $|v| \le 1 - \kappa$, the non-linearity ψ satisfies the sector inequalities

$$\alpha v^2 \le v\psi(v) \le \beta v^2$$

with

$$\alpha = -\frac{1-\kappa}{2-\kappa} \qquad\qquad \beta = \frac{1-\kappa}{\kappa}$$

Let $D(a, b)$ denote the circular disc in the complex plane, which is centered on the real axis, and with boundary intersecting the real axis as a and b. According to the circle criterion [113], a sufficient condition for stability of the feedback system is that the Nyquist curve $z = G(i\omega)$ stays inside the disc $D(-1/\beta, -1/\alpha)$, with a positive margin.

In our case, this disc is $D(-\kappa/(1-\kappa), (2-\kappa)/(1-\kappa))$, centered at $z = 1$, and has radius $r = 1/(1-\kappa) > 1$. To see that the Nyquist curve is inside this circle, with a positive margin, consider the smaller disc $D(1-r, 1)$, which is tangent to the original disk at $z = 1-r$, and tangent to the Nyquist curve at $z = 1$, illustrated in Fig 4.1. Define the linear fractional transformation $h(z)$ which maps $D(1-r, 1)$ onto the unit disc, and keeps the origin fixed,

$$h(z) = \frac{z}{(1-2\kappa)z + 2\kappa}$$

Under this transformation, illustrated in Fig 4.2, we find

$$\begin{aligned}h(G(i\omega)) &= \frac{\kappa e^{-i\omega}}{(1-2\kappa)\kappa e^{-i\omega} + 2\kappa(i\omega + \kappa e^{-i\omega})} = \frac{1}{1 + 2i\omega e^{i\omega}}\\ |h(G(i\omega))|^2 &= \frac{1}{|1 + 2i\omega e^{i\omega}|^2} = \frac{1}{1 + 4\omega(\omega - \sin\omega)} \le 1\end{aligned}$$

where the final inequality uses the bound $\sin\omega \le \omega$. We also see that the curve touches the unit circle only when $\omega = 0$. Hence, the Nyquist curve is inside $D(1-$

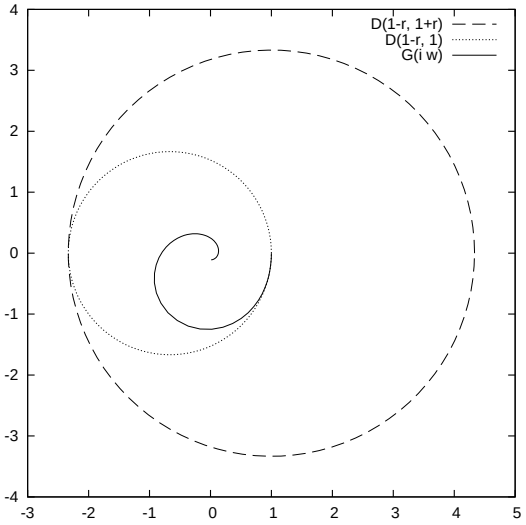

Figure 4.1: The Nyquist curve and the discs considered in the proof of Lemma 4.3, drawn for $\kappa = 0.8$. The large dashed circle corresponds to the sector inequalities for the nonlinearity ψ. The solid curve is the Nyquist curve for $G(s)$. The smaller circle is constructed so that it is tangent to both.

$r, 1)$, getting close only at 1 (for $\omega = 0$), and it follows that the Nyquist curve is inside $D(1-r, 1+r)$ with a positive margin. □

With this result, global asymptotic stability, regardless of initial state, follows as soon as we have established boundedness.

Lemma 4.4 *For each initial state ϕ_0 there exists a $T = T(\phi_0) \geq 0$ such that the unique solution of Eq.* (4.2) *satisfies*

$$0 < q(t) \leq 2q^* \tag{4.3}$$

for all $t \geq T$.

Proof: We first note the following upper bound for $\dot{q}(t)$:

$$\dot{q}(t) \leq q^*/\tau$$

valid for all $t \geq 0$.

(i) There exists $t_0 \geq \tau$ such that $q(t_0) \leq 2q^*$.

Assume to the contrary that no such t_0 exists; then $q(t) > 2q^*$ for all $t \geq \tau$. Since $q(t-\tau) > q^*$ implies that $\dot{q}(t) < 0$, we see that $\dot{q}(t) \leq 0$ for all $t \geq 2\tau$.

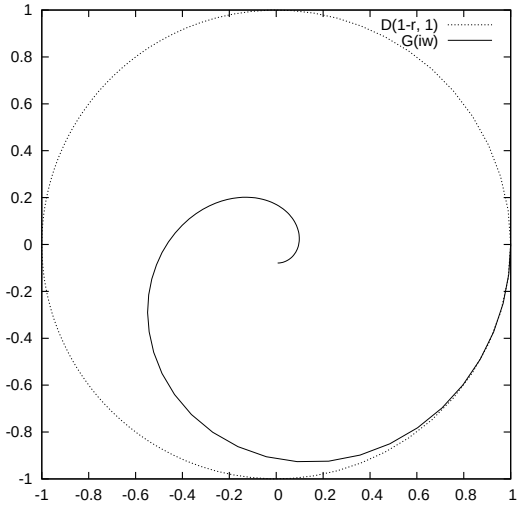

Figure 4.2: The Nyquist curve under the mapping $h(z)$, which maps the smaller circle from Fig 4.1 onto the unit circle, leaving the origin fixed.

This means that $q(t)$ is a decreasing function bounded below by $2q^*$, hence converging to some limit no smaller than $2q^*$. The limit value must be a stationary point, but the only stationary point is q^*. This contradiction proves the first step.

(ii) $q(t) \leq 2q^*$ for all $t \geq t_0$.

It is sufficient to prove that $q(t) \leq 2q^*$ for $t_0 \leq t \leq t_0+\tau$; then the rest follows by induction.

In case $q(t_0) \leq q^*$, the results follows directly from the bound on $\dot{q}$: for all $t_0 \leq t \leq t_0 + \tau$ we have

$$q(t) \leq q(t_0) + (t - t_0)q^*/\tau \leq 2q^*$$

Otherwise, $q(t_0) > q^*$. Put $q(t_0) = (1+\xi)q^*$, with $0 < \xi \leq 1$. Again, the key is the bound on $\dot{q}$. We have $q(t-\tau) \geq q(t_0)+(t-t_0-\tau)q^*/\tau = (\xi+(t-t_0)/\tau)q^* > 0$. To use this, express $q(t)$ in terms of the integral from t_0 to t,

$$q(t) = q(t_0) + \int_{t_0}^{t} \dot{q}(u)\,\mathrm{d}u = (1+\xi)q^* + \gamma \int_{t_0}^{t} \frac{q^* - q(u-\tau)}{\tau + q(u-\tau)/c}\,\mathrm{d}u$$

By substituting the bound for $q(u-\tau)$ in the numerator of the integrand, and replacing the integration interval by the set where the new integrand is

non-negative, we get

$$\begin{aligned} q(t) &\le (1+\xi)q^* + \gamma q^* \int_0^{t-t_0} \frac{1-\xi-u/\tau}{\tau + q(t_0+u-\tau)/c}\,\mathrm{d}u \\ &\le (1+\xi)q^* + q^* \int_0^{(1-\xi)\tau} \frac{1-\xi-u/\tau}{\tau + q(t_0+u-\tau)/c}\,\mathrm{d}u \\ &\le (1+\xi)q^* + q^* \int_0^{(1-\xi)\tau} \left(\frac{1-\xi}{\tau} - \frac{u}{\tau^2}\right)\mathrm{d}u \\ &= \left(1+\xi+\frac{(1-\xi)^2}{2}\right)q^* = \frac{3+\xi^2}{2}q^* \le 2q^* \end{aligned}$$

(iii) There exists a t_1 such that $q(t) > 0$ for all $t > t_1$.

Put $t_1 = t_0 + \tau$. By induction, it is sufficient to show that $q(t) > 0$ for $t_1 < t \le t_1 + \tau$. From the previous step we know that $q(t-\tau) \le 2q^*$ for all t in the interval $t_1 \le t \le t_1 + \tau$. This implies a lower bound on $\dot{q}(t)$ on the same interval. Recall that the function $f(q)$ is decreasing. Then

$$\dot{q}(t) \ge f(q(t)) = \gamma\frac{q^* - q(t-\tau)}{\tau + q(t-\tau)/c)} \ge -\gamma\frac{q^*}{\tau + 2q^*/c} > -\frac{q^*}{\tau}$$

(Without the upper bound on q, we have the lower bound $\dot{q} > -c$, which is not as useful). The result now follows using an argument similar to the previous step, but here we have strict inequalities. In case $q(t_1) \ge q^*$, it follows directly that

$$q(t) > q(t_1) - (t-t_1)q^*/\tau \ge 0$$

Otherwise, $q(t_1) < q^*$. Put $q(t_1) = (1-\xi)q^*$, with $0 < \xi \le 1$. We get $q(t-\tau) \le q(t_1)+(t_1-t+\tau)q^*/\tau = (2-\xi-(t-t_1)/\tau)q^*$, with equality only when $t = t_1+\tau$. It then follows that

$$\begin{aligned} q(t) = q(t_1) + \int_{t_1}^{t} \dot{q}(u) &= (1-\xi)q^* + \gamma\int_{t_1}^{t} \frac{q^* - q(u-\tau)}{\tau + q(u-\tau)/c}\,\mathrm{d}u \\ &> (1-\xi)q^* + \gamma q^* \int_0^{t-t_1} \frac{\xi - 1 + u/\tau}{\tau + q(t_1+u-\tau)/c}\,\mathrm{d}u \\ &\ge (1-\xi)q^* - \gamma q^* \int_0^{(1-\xi)\tau} \frac{1-\xi-u/\tau}{\tau + q(t_1+u-\tau)/c}\,\mathrm{d}u \\ &\ge (1-\xi)q^* - q^* \int_0^{(1-\xi)\tau} \left(\frac{1-\xi}{\tau} - \frac{u}{\tau^2}\right)\mathrm{d}u \\ &= \left(1-\xi-\frac{(1-\xi)^2}{2}\right)q^* = (1-\xi^2)\frac{q^*}{2} \ge 0 \end{aligned}$$

Hence, choose any $T > t_1$, and the lemma is true. □

Combining these Lemma 4.3 and 4.4, it follows that Theorem 4.2 holds.
Remarks:

(i) Since $\dot{q}(t) > -c$ for all $t \geq 0$, we can not hope for global exponential stability.

(ii) The strict inequality $q(t) > 0$ is essential, since it means that the non-negativity constraint in the system equation, Eq 4.2, does not affect the trajectory for $t \geq T$.

(iii) Actually, we see that the non-negativity constraint can get activated only in two ways: For $0 \leq t \leq \tau$, some initial conditions can force $q(t) = 0$. For $t \geq \tau$, the queue can get empty only if the initial queue is large and the capacity is large enough, but in this case $q(t) = 0$ is possible only during the first interval of length τ when $q(t) \leq 2q^*$.

Convergence time

We have seen that the system consisting of a single link and a single bottleneck is globally stable. When using cascade control, it is essential that the inner-loop is significantly faster than the outer-loop.

This section derives an upper bound for the time constant of the inner-loop. It can be used to aid the design of the window update mechanism of the outer-loop. The main result of the section can be summarized as, for arbitrary capacities and delays, transmission control based on the ACK-clock ensures that the sending rate and the bottleneck queue size converge within a small number of RTTs.

We concentrate on the convergence rate for small variations, using a linearization of the queue dynamics. Note that since the link serving the queue is of finite capacity, we can not expect to have exponential convergence for arbitrary large initial values.

The linearized dynamics were studied in Sec. 3.4, where we derived the transfer function

$$G(s) = \frac{e^{-s\tau} + \tau^* s}{\gamma e^{-s\tau} + \tau^* s}$$

The system poles are the solution to the equation

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

Let $z = \tau s$, and $\kappa = \gamma\tau/\tau^*$. Rewrite the equation as

$$z = -\kappa e^{-z} \tag{4.4}$$

The poles can be expressed as $p_j = W_j(-\kappa)$, where j enumerates the branches of Lambert's function. Figure 4.3 shows the first few poles, $-3 \leq j \leq 2$, for some values of κ. Since the system is stable, we know that $\operatorname{Re} z < 0$ for all solutions to (4.4). Numerically, it appears that the pole $W_0(-\kappa)$ is the one closest to the real axis, and hence dominating for the dynamics. But since we do not have a strict proof for that, our next goal is instead to find a bound σ such that $\operatorname{Re} z \leq -\sigma$ for all

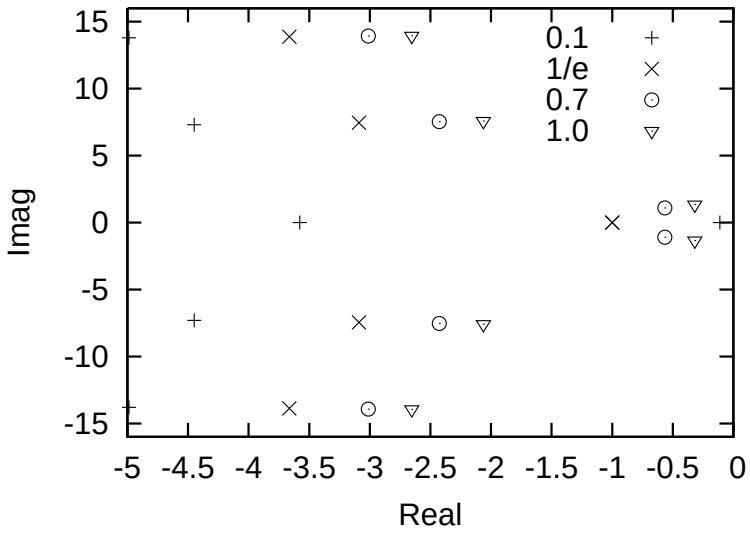

Figure 4.3: The first seven system poles, for $\kappa = 0.1$, $1/e$, 0.7, and 1.0. For $\kappa = 1/e$, there is a double pole at -1.

solutions to (4.4). Then all transients of the system are bounded by the exponential $Ce^{-\sigma t/\tau}$, which means that the system's convergence time constant is at most τ/σ.

Assume that $z = -x + iy$, $x > 0$, is a pole of the system. Substitution into (4.4) yields

$$-x + iy = z = -\kappa e^{-z} = -\kappa e^{x} e^{-iy} \tag{4.5}$$

First, we derive a bound for $|y|$ in terms of x. We have

$$|y| < |z| = \kappa e^{x}$$

Assume that $x \leq \log(\pi/(2\kappa))$, so that both left and right hand side of this equation lie in the interval $[0, \pi/2]$, where the cos function is decreasing. Then

$$\cos|y| > \cos(\kappa e^{x})$$

Next, use this to bound the real part of Equation (4.5),

$$x = \kappa e^{x} \cos y > \kappa e^{x} \cos(\kappa e^{x}) \tag{4.6}$$

Define σ as the smallest positive solution to

$$u = \kappa e^{u} \cos(\kappa e^{u})$$

Since the right hand side is zero for $u = \log(\pi/(2\kappa))$, it follows by continuity that $\sigma < \log(\pi/(2\kappa))$. Hence $x \leq \sigma$ leads to a contradiction, and it follows that $\operatorname{Re} z < -\sigma$ for all solutions to (4.4).

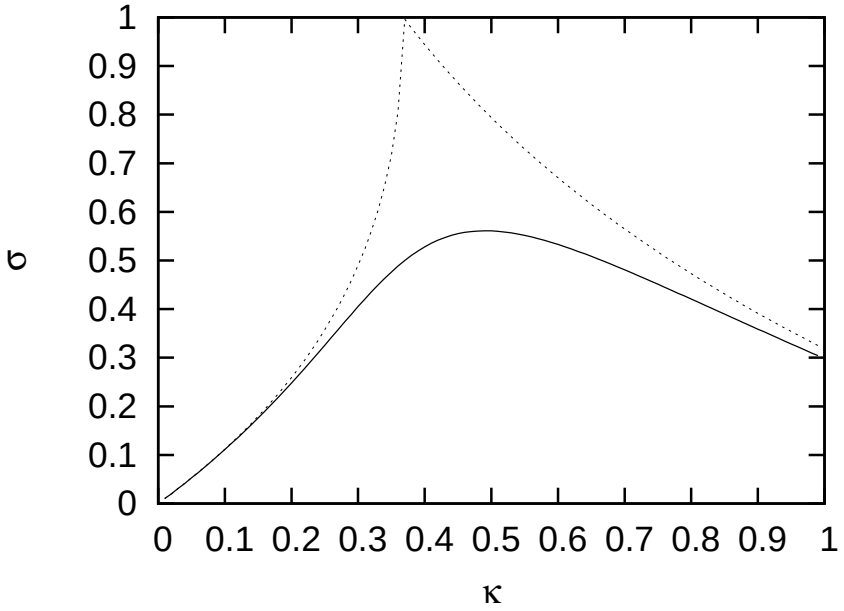

Figure 4.4: The solid curve is the bound σ, as a function of κ. The dotted curve is $-\operatorname{Re} W_0(-\kappa)$, i.e., the distance from the imaginary axis to the pole corresponding to the W_0 branch of Lambert's function.

Figure 4.4 shows σ as a function of κ. In the same figure, the dotted curve also shows the negative real part of the pole associated with the W_0 branch. If $\kappa \geq 1/4$, e.g., if $\gamma \geq 1/2$ and $\tau/\tau^* \geq 1/2$, then $\sigma \geq 0.298$, and the system time constant is at most 3.35τ. These conditions correspond to a typical network configuration where queueing delay is no greater than the propagation delay, and at most half of the capacity is used by inelastic cross-traffic.

When $\kappa \to 0$, also $\sigma \to 0$ and the time constant tends to infinity. Let us examine the two factors in $\kappa = \gamma\tau/\tau^*$. When $\gamma \to 0$, we can expect the system to become very slow, no matter how we measure it. However, when τ/τ^* is small, i.e., when the queueing delay is large in comparison to the propagation delay, and $\tau^* \gg \tau$, it makes sense to measure the time constant in terms of τ^*. The convergence time constant is bounded by τ^*/μ, where μ is defined by

$$\mu = \frac{\sigma\tau^*}{\tau}$$

Figure 4.5 shows μ, as a function of τ/τ^*, for some different values of γ. Note that for small values of γ, the minimum is attained for $\tau/\tau^* = 0$, i.e., when the queueing delay approaches infinity. Figure 4.6 shows the minimum μ as a function of γ. Note

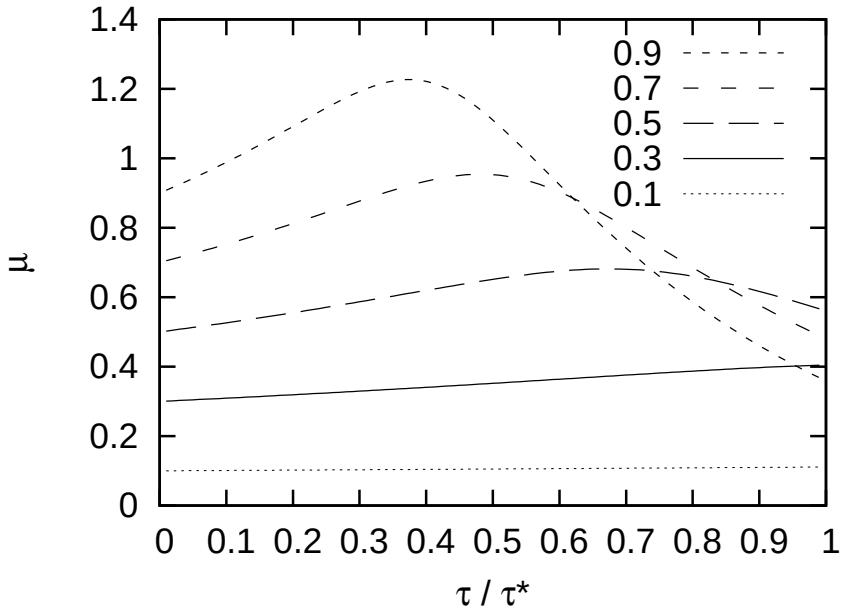

Figure 4.5: For the five values $\gamma = 0.1$, 0.3, 0.5, 0.7, and 0.9, this figure shows μ as a function of τ/τ^*. The value of μ is related to the time constant expressed in units of τ^*.

that this curve is linear, $\mu \approx \gamma$, for small γ.

We summarize these findings as a proposition.

Proposition 4.5 (Convergence time) *Consider the system*

$$\dot{q}(t) = \frac{w}{\tau + q(t-\tau)/c} - \gamma c$$

where $c > 0$ is the link capacity, $\tau > 0$ the queueing delay, w is a constant window size, and the parameter γ is the proportion of the capacity which is not used by cross-traffic. Assume that $w > c\tau$, so that the equilibrium queue size q^* is positive,

$$q^* = \frac{w}{\gamma} - c\tau > 0.$$

Also define the equilibrium RTT *$\tau^* = \tau + q^*/c$, and put $\kappa = \gamma\tau/\tau^*$. Let T denote the time constant of the linearized system close to the equilibrium. Then the following holds.*

(i) If $\kappa > 1/4$, which is a typical network configuration, then $T < 3.4\tau$.

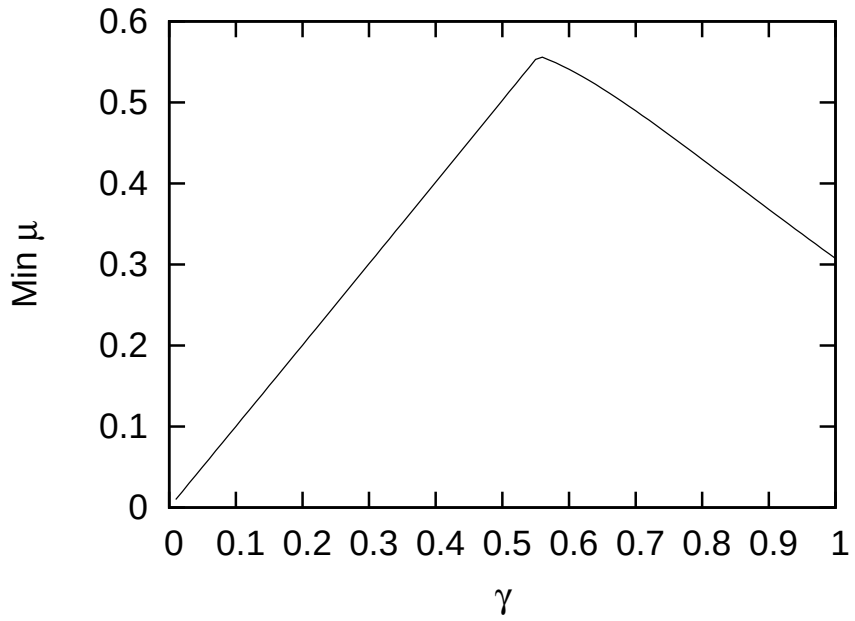

Figure 4.6: The minimum value of μ, as a function of γ. Note that the function is linear for small γ.

(ii) If $\gamma \leq 1/2$, then $T < \tau^*/\gamma$. This bound is actually tight for small γ, since one of the poles is located at $W_0(-\gamma\tau/\tau^*)/\tau = -\gamma/\tau^* + O(\gamma^2)$ (see Sec. 3.4).

(iii) For all a $\gamma > 0.3$, $T < 3.4\tau^*$.

When $\gamma > 0.3$, convergence time is always bounded by $3.4\tau^*$, the total RTT including queueing. Furthermore, for typical configurations, the same bound is valid also if τ^* is replaced by τ, the propagation roundtrip delay. In the extreme case of small γ, the convergence time is large, inversely proportional to γ.

For typical network paths, with an RTT of at most a few 100 ms, we can expect convergence time on the order of 1 s. This can be compared to the convergence time of proposed outer-loops, such as TCP with AQM, with a convergence time of 20–60 s [56], and Fast TCP, with convergence times up to several minutes [115].

Let us compare this time constant bound to our earlier simulation experiments, in Fig. 3.8. In these scenarios, we had an available capacity $\gamma c = 1$ Mbit/s and a propagation roundtrip delay of 50 ms. At $t = 10$ s, the window size was increased by five packets, from an initial value corresponding to queue size close to 20 packets. the new equilibrium queue size is $5/\gamma$ packets.

For the scenario with $\gamma = 0.2$, we have $\kappa = 0.11$ (computed for the equilibrium queue size and delay after the window change). For the time constant bound, we

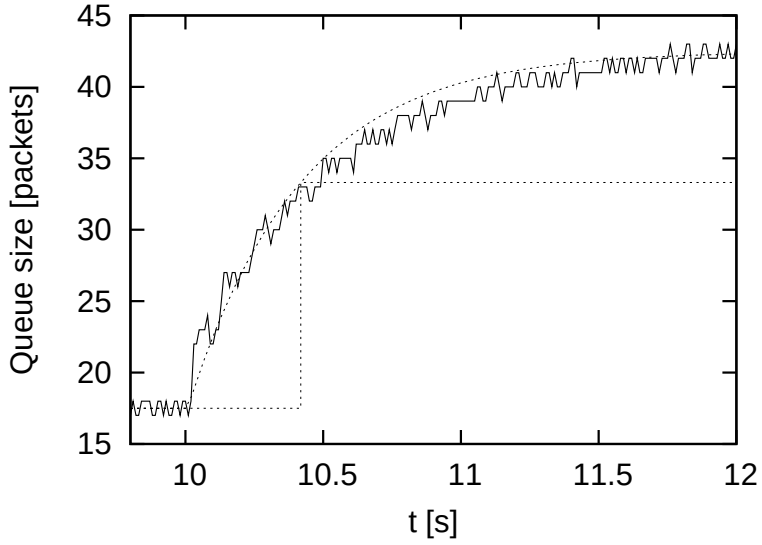

Figure 4.7: Queue response to a step change in the window size, for $\gamma = 0.2$. The dotted curves illustrate a first order filter with the predicted time constant 0.41 s.

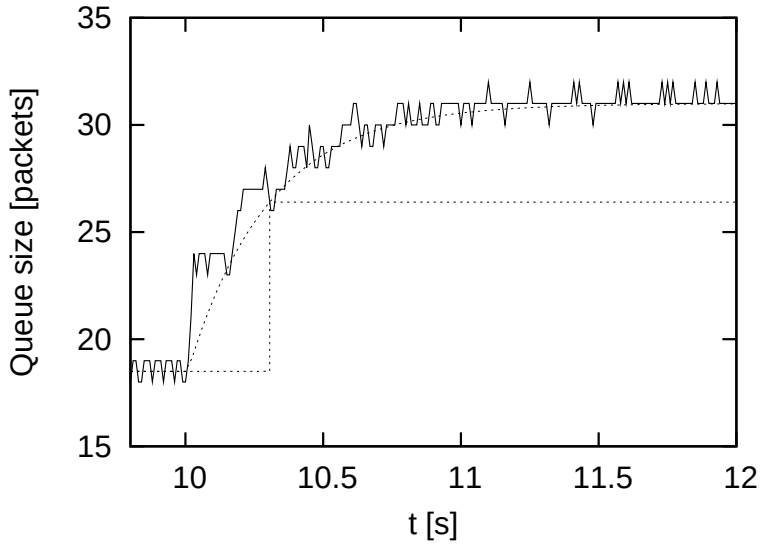

Figure 4.8: Queue response to a step change in the window size, for $\gamma = 0.4$. The dotted curves illustrate a first order filter with the predicted time constant 0.30 s.

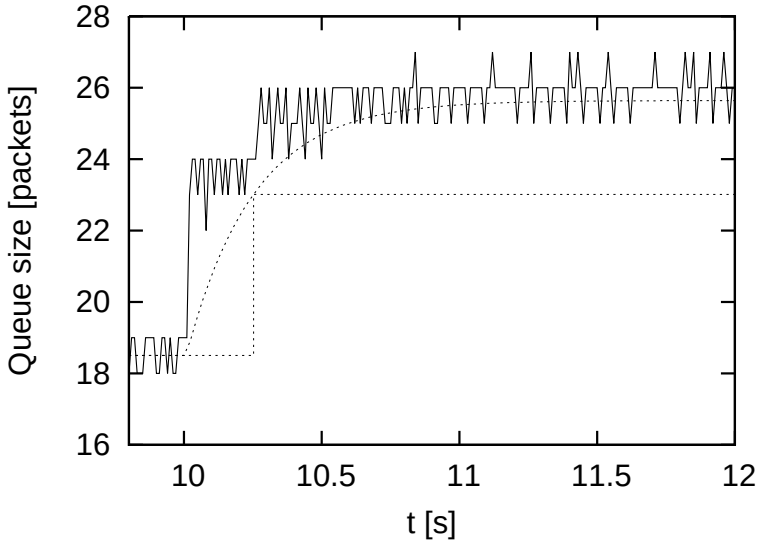

Figure 4.9: Queue response to a step change in the window size, for $\gamma = 0.7$. The dotted curves illustrate a first order filter with the predicted time constant 0.24 s.

get $\sigma/\tau = 0.41$ s. Figure 4.7 compares the predicted time constant to the `ns2` simulation. The bound σ/τ is very close to the actual convergence speed.

For the scenarios with $\gamma = 0.4$ and $\gamma = 0.7$ we get a predicted convergence time of 0.30 s and 0.24 s, respectively. These are illustrated in Figs. 4.8 and 4.9. When γ gets close to 1, the bound becomes very conservative. The reason is that σ is computed from the system poles only, and we get pole-zero cancellation when γ approaches 1. For $\gamma = 1$, the joint link model reduces to the static link model, with no dynamics, and then q converges within a single RTT.

4.3 Multiple flows

So far, we have considered a single flow subject to window-based flow control, sharing a single bottleneck link with some constant rate cross-traffic. A large proportion of such cross-traffic leads to a larger static gain from window size to queue size, and larger time constants. What if some of that "cross-traffic" is also using window-based congestion control? That is the scenario considered in this section.

We have n flows, each with a constant window size w_k and a roundtrip propagation delay τ_k. They share a bottleneck of capacity c with cross-traffic at a constant rate $(1-\gamma)c$. The model in Sec. 3.6 gives the following queue dynamics:

$$\dot{q}(t) = \sum_k \frac{w_k}{\tau_k + q(t-\tau_k)/c} - \gamma c$$

We also have a non-negativity constraint, forcing $q(t) \geq 0$.

Theorem 4.6 *For arbitrary $w_k > 0$, $\tau_k > 0$, $c > 0$ and $0 < \gamma \leq 1$, the dynamical system*

$$\dot{q}(t) = \begin{cases} \sum_k \dfrac{w_k}{\tau_k + q(t-\tau_k)/c} - \gamma c & q(t) > 0 \\ \max\left(0, \sum_k \dfrac{w_k}{\tau_k + q(t-\tau_k)/c} - \gamma c\right) & q(t) = 0 \end{cases}$$

has a unique equilibrium $q^* \geq 0$, which is locally stable.

Proof: Start with the simplest case. Assume that

$$\sum \frac{w_k}{\tau_k} \leq \gamma c$$

Then $q^* = 0$ is the unique equilibrium of the system. Furthermore, $\dot{q}(t) \leq 0$ for all t. Hence $q(t)$ is a non-increasing function bounded below by zero. It must converge to some limit as $t \to \infty$. The limit must be an equilibrium of the system, hence it is zero.

The remaining, more interesting, case is that $\sum w_k/\tau_k > \gamma c$. Then the system has a unique equilibrium $q^* > 0$ characterized by

$$\sum_k \frac{w_k}{\tau_k + q^*/c} = \gamma c$$

We then have

$$\begin{aligned} \dot{q}(t) &= \sum_k \frac{w_k}{\tau_k + q(t-\tau_k)/c} - \gamma c \\ &= \sum_k \left(\frac{w_k}{\tau_k + q(t-\tau_k)/c} - \frac{w_k}{\tau_k + q^*/c} \right) \\ &= \sum_k \frac{w_k\big(q^* - q(t-\tau_k)\big)/c}{\big(\tau_k + q(t-\tau_k)/c\big)\big(\tau_k + q^*/c\big)} \end{aligned}$$

Now put $q(t) = q^* + \widetilde{q}(t)$ and linearize, to get

$$\dot{\widetilde{q}}(t) = -\sum_k \frac{cw_k}{(c\tau_k + q^*)^2} \widetilde{q}(t-\tau_k)$$

We now use Theorem 4.17, which says that $\dot{x}(t) = -\sum \alpha_k x(t-\tau_k)$ is stable provided $\sum \alpha_k \tau_k \leq 1$. Put $\alpha_k = cw_k/(c\tau_k + q^*)^2$. We have

$$\sum_k \alpha_k \tau_k = \sum_k \underbrace{\frac{c\tau_k}{c\tau_k + q^*}}_{<1} \frac{w_k}{c\tau_k + q^*} < \sum_k \frac{w_k}{c\tau_k + q^*} = \gamma \leq 1$$

By Theorem 4.17, the linearized system is stable, and hence the non-linear system is locally stable. □

4.4 Stability for a general network topology

Now consider a general network topology, described by a routing matrix A (See Sec. 2.3). Each source uses a fixed window size w_k and has a roundtrip propagation delay $\tau_s^0 > 0$. For the links, we use the *queueing delay* as state variables, denoted d_ℓ, rather than the queue size. Link capacities are denoted c_ℓ.

For the sending rate at each source, we use the simplified model without signalling delays,

$$r_s(t) = \frac{w_s}{\tau_s^0 + \tau_s(t)}$$

where $\tau = A^T d$ is the aggregated queueing delay.

At each link, the aggregated arrival rate is $y = Ar$ (this ignores the issue that a sources sending rate may be throttled at an earlier bottleneck). The queueing delay is $d_\ell = q_\ell / c_\ell$, and the differential equation for d is

$$\dot{d}_\ell(t) = \begin{cases} \frac{y_\ell(t)}{c_\ell} - 1 & d_\ell(t) > 0 \\ \max\left(0, \frac{y_\ell(t)}{c_\ell} - 1\right) & d_\ell(t) = 0 \end{cases}$$

Theorem 4.7 *Assume that $w > 0$, $c > 0$, and that A has full row rank. Then the queue dynamics are globally asymptotically stable.*

Proof: Define the function

$$V(d) = c^T d - \sum_s w_s \log(\tau_s^0 + A_s^T d)$$

where A_s denotes the column of A corresponding to source s, so that $A_s^T d = \tau_s$. Then the gradient of V is

$$\nabla V = c - \sum_s \frac{w_s}{\tau_s^0 + A_s^T d} A_s = c - \sum_s A_s r_s = c - Ar = c - y$$

and the Hessian matrix is

$$H(V) = \sum_s \frac{w_s}{(\tau_s^0 + A_s^T d)^2} A_s A_s^T$$

The assumption that A has full rank implies that $H(V)$ is positive definite. Too see this, examine that $u^T H(V) u$ for an arbitrary vector u. We get

$$u^T H(V) u = \sum_s \underbrace{\frac{w_s}{(\tau_s^0 + A_s^T d)^2}}_{>0} \left(A_s^T u\right)^2 \geq 0$$

So $H(V)$ is clearly positive semidefinite. To see that it is positive definite, assume that $u^T H(V) u = 0$. Then $A_s^T u = 0$ for each s, or $A^T u = 0$. Since A has full row rank, this implies that $u = 0$.

Since $H(V)$ is positive definite, it is clear that V is a strictly convex function, on the convex set $d \geq 0$. Since $V(d)$ is bounded from below, and $V(d) \to \infty$ when $d \to \infty$, V has a unique global minimizer d^*.

Since d^* is a minimizer d, then for each ℓ, either $d^*_\ell = 0$ and $\partial V(d^*)/\partial d_\ell > 0$ or $\partial V(d^*)/\partial d_\ell = 0$. This is equivalent to the equilibrium condition, $y_\ell \leq c_\ell$ and $d_\ell(c_\ell - y_\ell) = 0$. Hence d^* is an equilibrium for the dynamical system.

We also have

$$\frac{\mathrm{d}V(d(t))}{\mathrm{d}t} = \nabla V \cdot \dot{d} = \sum_{\substack{\ell \\ d_\ell > 0 \\ \text{or } y_\ell > c_\ell}} -(c_\ell - y_\ell)^2 \leq 0$$

with strict inequality for all $d \neq d^*$. By the strict convexity of V, it follows that the function $V(d) - V(d^*)$ is a Lyapunov function for the system [67], and it follows that the equilibrium d^* is globally asymptotically stable. □

4.5 Differential equations with state constraints

In this and coming chapters, we use differential equations of the form

$$\begin{aligned} \dot{x}_k(t) &= \begin{cases} f_k(t, x(t)) & x_k(t) > 0 \\ \max(0, f_k(t, x(t))) & x_k(t) = 0 \end{cases} \\ x_k(0) &= x^* \end{aligned}$$

as well as the time-delayed differential equation

$$\begin{aligned} \dot{x}_k(t) &= \begin{cases} f_k(t, x(t-1)) & x_k(t) > 0 \\ \max(0, f_k(t, x(t-1))) & x_k(t) = 0 \end{cases} \\ x_k(t) &= \phi_k(t) \quad \text{for } -1 \leq t \leq 0 \end{aligned}$$

The state x is constrained to the set $x_k \geq 0$ for all k. In this section, we prove existence and uniqueness of solutions for differential equations of these forms. Even if the functions f_k are very regular, the right hand side has a discontinuity at the boundary, which violates the Lipschitz condition in standard proofs of existence and uniqueness of solutions.

One way to think about such a system is as a hybrid system, where the discrete state specifies which of the constraints are active, i.e., at each time t, the discrete state gives the set of k such that $x_k = 0$ and $f_k < 0$. There are some drawbacks with this approach:

- The number of discrete states is exponential, 2^n, where n is the number of continuous states, i.e., the dimension of x.

- Let t_k be the sequence of switching times, i.e., times when the set of active constraints change. This sequence may have limit points, so called Zeno behavior. It is not obvious if limit points can be excluded, or what additional requirements one would have to place on the functions f_k and, in the time delayed case, on the initial condition.

- It is difficult, given the state at a time t, to determine which constraints really are active. In the borderline case that both x_k and f_k are zero, the corresponding constraint may or may not be active.

The main idea of our proofs is to construct candidate solutions by ignoring the constraints, and use a projection procedure to ensure that the constraints are satisfied. The projection procedure is carefully defined so that it does not try to enumerate the intervals where the constraints are active, thereby avoiding any problems with limit points or Zeno behavior. In particular for the time-delayed equation, trajectories like $x(t) = (t-1/2)^2 \max(0, \sin^2(1/(t-1/2))$ may be possible for some initial condition ϕ. This trajectory is continuously differentiable, still zero on infinitely many intervals, accumulating at $t = 1/2$.

Right-hand derivatives

The solutions to the ordinary differential equations are continuously differentiable. But for differential equations with state constraints, we need a slightly relaxed notion of "solution". A solution is required to be continuous and right-hand differentiable for all $t \geq 0$. In this section, we use the convention that $\mathrm{d}/\,\mathrm{d}t$ and the dot symbol denotes right-hand differentiation.

The right hand derivative of a function $x(t)$ is defined by

$$\dot{x}(t) = \lim_{h \to 0+} \frac{x(t+h) - x(t)}{h}$$

when this limit exists. The definition has meaning only for t such that x is defined on the interval $[t, t+\epsilon]$ for some $\epsilon > 0$. E.g,. if x is defined on the interval $a \leq t \leq b$, the limit makes no sense at the right edge, $t = b$, but only for $a \leq t < b$. When we say that $\dot{x}$ exists for all t, we mean that the limit exists at all t where it has meaning. Existence of this limit implies that

$$x(t+h) = x(t) + (\dot{x}(t) + r(h))\, h$$

where $r(h) \to 0$ as $h \to 0+$.

Since the properties of right-hand differentiable functions are usually not treated in any detail in standard calculus books, we state and prove the most important properties here. We assume that all functions are defined on some finite or infinite interval, that they are continuous, and that the right-hand derivative exists for all t. These assumptions are not repeated for each proposition.

Proposition 4.8 *If $\dot{x}(t) > 0$ for all t, then $x(t)$ is strictly increasing.*

Proof: Let t_1 and t_2 be arbitrary, with $t_2 > t_1$, and consider the restriction of x to the closed interval $t_1 \leq t \leq t_2$. Since x is continuous, it has a maximum value on this interval. Let $M = \max x(t)$. For all t, $t_1 \leq t < t_2$, and all small enough h, we have

$$x(t+h) = x(t) + \big(\dot{x}(t) + r(h)\big)\, h > x(t)$$

This shows that $x(t) < x(t+h) \leq M$. The only point where M can be achieved is therefore the right end point, so we have

$$x(t_2) = M > x(t)$$

for all $t_1 \leq t < t$. In particular, $x(t_2) > x(t_1)$, which concludes the proof. □

Proposition 4.9 *If $\dot{x}(t) \geq 0$ for all t, then $x(t)$ is increasing.*

Proof: Let $\epsilon > 0$. Put $y(t) = x(t) + \epsilon t$. Then $\dot{y}(t) = \dot{x}(t) + \epsilon > 0$, and it follows that y is strictly increasing. Consider two arbitrary points $t_2 > t_1$. Then

$$x(t_2) - x(t_1) = y(t_2) - \epsilon t_2 - y(t_1) + \epsilon t_1 > -\epsilon(t_2 - t_1).$$

Since ϵ was arbitrary, we must have $x(t_2) - x(t_1) \geq 0$. □

Proposition 4.10 *If $\dot{x}(t) = 0$, on some interval, then $x(t)$ is constant on that interval.*

Proof: Both $x(t)$ and $-x(t)$ are increasing. □

Proposition 4.11 (Mean value theorem) *Let $t_2 > t_1$. Then there exists a ξ, with $t_1 \leq \xi < t_2$, such that*

$$\dot{x}(\xi) \geq \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

Proof: Define

$$K = \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

Assume that no such ξ exists. Then $\dot{x}(t) < K$ for all $t_1 \leq t < t_2$. Put $y(t) = K\,(t - t_1) - x(t)$. Then $\dot{y}(t) > 0$ on the interval, and y is strictly increasing. But we also have $y(t_1) = y(t_2) = -x(t_1)$. This contradiction concludes the proof. □

A projection operator

We will use the following projection operator P. Let ϕ be a continuous function on some interval $0 \leq t \leq T$, with $\phi(0) \geq 0$. Assume that ϕ has bounded descent,

$$0 \leq t \leq s \leq T \implies \phi(s) - \phi(t) \geq -M\,(s-t)$$

for some constant M. Let S_ϕ be the set of continuous functions u such that

(i) $u(0) \geq 0$

(ii) $0 \leq t \leq s \leq T \implies 0 \leq u(s) - u(t) \leq M\,(s-t)$

(iii) $\phi(t) + u(t) \geq 0$ for all $0 \leq t \leq T$.

Since $M\,t \in S_\phi$, the set is non-empty. The functions are also bounded below, by zero. Then the point-wise infimum is well defined, and we can define

$$u_\phi(t) = \inf_{u \in S_\phi} u(t)$$

The bound (ii) implies that u_ϕ is continuous. By the minimality of u_ϕ, it also follows that u_ϕ depends only on ϕ, not on the particular value of M.

We can now define the projection operator P by

$$(P\phi)(t) = \phi(t) + u_\phi(t)$$

Lemma 4.12 *The projection operator P has the following properties.*

(i) $P(\phi)(t) \geq 0$

(ii) $P(\phi)(0) = \phi(0)$

(iii) $\|P(\phi) - P(\psi)\| \leq 2\|\phi - \psi\|$, *where $\|\cdot\|$ denotes the supremum norm.*

Proof: (i) follows directly from the definition of S_ϕ. For (ii), note that $M\,t \in S_\phi$. Then $u_\phi(0) = 0$ by minimality.

To prove (iii), define S_ϕ and S_ψ using some common, large enough, M. Since

$$\psi(t) + \|\phi - \psi\| + u_\phi(t) \geq \phi(t) + u_\phi(t) \geq 0$$

we see that $u_\phi + \|\phi - \psi\| \in S_\psi$. Hence $u_\psi(t) \leq u_\phi(t) + \|\phi - \psi\|$ for all t. By switching the roles of ϕ and ψ, we find that

$$\begin{aligned} \|u_\phi - u_\psi\| &\leq \|\phi - \psi\| \\ |P(\phi)(t) - P(\psi)(t)| &= \big|\phi(t) + u_\phi(t) - (\psi(t) + u_\psi(t))\big| \\ &\leq \|\phi - \psi\| + \|u_\phi - u_\psi\| \\ &\leq 2\|\phi - \psi\| \end{aligned}$$

Since this is valid for all t, (iii) follows. □

Lemma 4.13 *Assume that ϕ is continuous, and equal to the integral of some right-hand continuous and integrable function f,*

$$\phi(t) = \phi_0 + \int_0^t f(s)\, ds$$

Also assume that f is bounded below, $f(t) \geq -M$. Then

(i) $u_\phi(t+h) - u_\phi(t) \leq \int_t^{t+h} |f(s)|\, ds$ *for* $h > 0$.

(ii) $P(\phi)$ *is right-hand differentiable, with*

$$\frac{d}{dt} P(\phi)(t) = \begin{cases} f(t) & P(\phi)(t) > 0 \\ \max(0, f(t) & P(\phi)(t) = 0 \end{cases} \tag{4.7}$$

Proof: (i) follows from the minimality of u_ϕ. To prove (ii), put $\psi = P(\phi) = \phi + u_\phi$ and consider an arbitrary t. There are a couple of different cases, related to the non-negativity constraint $\psi \geq 0$.

- If $\psi(t) > 0$, then it follows from continuity of ψ and the minimality of u_ϕ that u_ϕ is constant in a neighborhood of t. Then $\dot{u}_\phi(t) = 0$ and $\dot{\psi}(t) = f$.

- If $\psi(t) = 0$ and $f > 0$, then the right-hand continuity of f imply that ϕ is strictly increasing in some non-empty interval $[t, t+\epsilon]$. It follows from the minimality of u_ϕ that u_ϕ is constant in this interval, and hence its right-hand derivative is zero. So also in this case, $\dot{\psi}(t) = f(t)$, recalling that the dot denotes right-hand differentiation.

- If $\psi(t) = 0$ and $f < 0$, then ϕ is strictly decreasing in some interval $[t, t+\epsilon]$. Minimality of u_ϕ now implies that $\psi(t)$ is zero in this interval, and hence the right-hand derivative is zero.

- Finally, consider the case $\psi(t) = 0$ and $f = 0$. By (i),

  $$\frac{u_\phi(t+h) - u_\phi(t)}{h} \leq \frac{1}{h} \int_t^{t+h} |f(s)|\, \mathrm{d}s$$

  for $h > 0$. Let $h \to 0$ from the right. By the right-hand continuity of f, the limit is $f(t) = 0$, and hence $\dot{u}_\phi(t) = 0$ and also $\dot{\psi}(t) = 0$.

It follows that the right-hand derivative exists, for all t, and that Eq. (4.7) is valid. □

Existence and uniqueness

We can now prove existence and uniqueness. We first consider the special case that the right-hand side depends on the state only via the non-negativity condition.

Theorem 4.14 *Assume that $f(t)$ is measurable, bounded, and right-hand continuous for $0 \leq t \leq T$. Assume that $x_0 \geq 0$. Then the exists a unique continuous and right-hand differentiable $x(t)$ such that*

$$\dot{x}(t) = \begin{cases} f(t) & x(t) > 0 \\ \max(0, f(t)) & x(t) = 0 \end{cases}$$
$$x(0) = x_0$$

Proof: First define the candidate solution

$$\widetilde{x}(t) = x_0 + \int_0^t f(s)\,\mathrm{d}s$$

By the assumptions on f, $\widetilde{x}$ is clearly continuous and right-hand differentiable[1]. If it also happens to be non-negative, we can set $x(t) = \widetilde{x}(t)$. In the general case, let

$$x = P\widetilde{x} = \widetilde{x} + u_{\widetilde{x}}$$

It is clear that $x(0) = x_0$. An application of Lemma 4.13 shows that $x(t)$ satisfies the differential equation. To prove uniqueness, assume that y is any solution to the differential equation. Define

$$u(t) = y(t) - \widetilde{x}(t)$$

By assumption, $\dot{y}$ (the right-hand derivative) exists at all t. Then also $\dot{u}$ exists, and

$$\dot{u}(t) = \dot{y}(t) - \dot{\widetilde{x}}(t) = \begin{cases} 0 & y(t) > 0 \\ \max(-f(t), 0) & y(t) = 0 \end{cases}$$

Then $0 \leq \dot{u}(t) \leq M$ for all t. Since we also have $u(0) = 0$ and $\widetilde{x} + u = y \geq 0$, it follows that $u \in S_{\widetilde{x}}$, and hence $u(t) \geq u_{\widetilde{x}}(t)$ for all t. Consider the difference

$$d(t) = y(t) - x(t) = u(t) - u_{\widetilde{x}}(t) \geq 0$$

Assume that we have strict inequality at some point, $d(t_1) > 0$, with $t_1 > 0$. Apply the mean value theorem, Prop. 4.11, to the function $d(t)^2$. It follows that there exists a ξ, with $0 \leq \xi < t_1$ such that

$$2d(\xi)\dot{d}(\xi) \geq \frac{d(t_1)^2}{t_1} > 0$$

[1]The assumption that f is bounded above is needed only to ensure that $\widetilde{x}$ is continuous. The assumption that f is bounded below was needed to show that S_ϕ is non-empty, and that the infimum u_ϕ is continuous.

Since we know that $d(\xi) \geq 0$, it follows that $d(\xi) > 0$ and $\dot{d}(\xi) > 0$. It now follows that $y(\xi) = x(\xi) + d(\xi) > 0$, so that the inequality constraint is not active at $t = \xi$. Hence

$$\dot{y}(\xi) = f(\xi)$$

But on the other hand, $\dot{d}(\xi) > 0$ implies that $\dot{u}(\xi) > 0$, so that

$$\dot{y}(\xi) = \dot{\widetilde{x}}(\xi) + \dot{u}(\xi) > f(\xi)$$

This contradiction proves that $y(t) = x(t)$ for all t. □

We are now ready to handle the time-delayed system.

Theorem 4.15 *Assume that f_k are continuous and that ϕ_k are non-negative, integrable, bounded, and right-hand continuous. Then there exists a unique x such that*

$$\dot{x}_k(t) = \begin{cases} f_k(t, x(t-1)) & x_k(t) > 0 \\ \max(0, f_k(t, x(t-1))) & x_k(t) = 0 \end{cases}$$

for $t \geq 0$, with initial condition $x_k(t) = \phi_k(t)$ for $-1 \leq t \leq 0$.

Proof: It is clearly sufficient to show that there exists a unique solution for $0 \leq t \leq 1$ which is continuous and non-negative; then existence and uniqueness for all $t \geq 0$ follows by induction. Furthermore, on this interval, the differential equation can be re written as

$$\dot{x}_k(t) = \begin{cases} f_k(t, \phi(t-1)) & x_k(t) > 0 \\ \max(0, f_k(t, \phi(t-1))) & x_k(t) = 0 \end{cases}$$
$$x_k(0) = \phi_k(0)$$

This means that there is no coupling between the different states, except via the given initial condition ϕ. Furthermore, the assumptions on f_k and ϕ_k imply that the functions $f_k(t, \phi(t-1))$ are bounded, integrable, and right-hand continuous on the interval $0 \leq t \leq 1$. Hence Lemma 4.14 can be applied to each x_k, and existence and uniqueness on the interval $0 \leq t \leq 1$ follows. □

To handle the equation without time delay is a little more complicated, since we have a direct coupling between the different states. The idea of the proof is to use a Picard iteration with a projection in each step. Define the function sequence $\phi^{(j)}$ by

$$\phi_k^{(0)}(t) = x_k^*$$
$$\widetilde{\phi}_k^{(j+1)}(t) = x_k^* + \int_0^t f_k\big(\phi^{(j)}(s)\big)\,\mathrm{d}s$$
$$\phi^{(j+1)} = P\big(\widetilde{\phi}^{(j+1)}\big)$$

The theorem below shows that the mapping $\phi^{(j)} \mapsto \phi^{(j+1)}$, on a suitable interval, is a contraction, and that the fix-point is the unique solution to the given differential equation.

Theorem 4.16 *Assume that f_k are continuous in t and Lipschitz continuous in x, i.e., there exists some constant C such that $|f_k(t,x) - f_k(t,y)| \leq C \max_\ell |x_\ell - y_\ell|$ for all x, y, t, and k. Then, for any initial condition x^*, with $x_k^* \geq 0$, there exists a unique x such that*

$$\begin{aligned} \dot{x}_k(t) &= \begin{cases} f_k(t, x(t)) & \text{if } x_k(t) > 0 \\ \max(0, f_k(t, x(t))) & \text{if } x_k = 0 \end{cases} \\ x_k(0) &= x^* \end{aligned}$$

for all $t \geq 0$

Proof: Select $T = 1/(3C)$. It is clearly sufficient to prove that the there exists a unique solution for $0 \leq t \leq T$; since T does not depend on t or on the initial condition, existence and uniqueness for all $t \geq 0$ follows by induction.

We use the Banach space V of continuous functions mapping the interval $0 \leq t \leq T$ into $\{x \in R^n; x_k \geq 0\}$, using the supremum norm defined by

$$\begin{aligned} |\phi(t)|_\infty &= \max_k |\phi_k(t)| \\ \|\phi\| &= \sup_t |\phi(t)|_\infty = \sup_{t,k} |\phi_k(t)| \end{aligned}$$

We define an operator F mapping V into itself. $\psi = F(\phi)$ is defined by letting ψ_k be the uniquely defined solution to

$$\begin{aligned} \dot{\psi}_k(t) &= \begin{cases} f_k(t, \phi(t)) & \psi_k(t) > 0 \\ \max(0, f_k(t, \phi(t))) & \psi_k(t) = 0 \end{cases} \\ \psi_k(0) &= x_k^* \end{aligned}$$

Since ϕ and f_k are assumed continuous, the function $f_k(t, \phi(t))$ is continuous and hence integrable and bounded on the closed interval $0 \leq t \leq T$. By Lemma 4.14, F is well defined. Recalling the construction in that proof, F can be decomposed as $F(\phi) = P(G(\phi))$, where G is the usual Picard iteration operator,

$$G_k(\phi)(t) = x_k^* + \int_0^t f_k(\phi(s))\, \mathrm{d}s$$

and P is the projection operator in Lemma 4.12, applied independently to each $G_k(\phi)$.

Since $G_k(\phi)$ is differentiable, with

$$\frac{\mathrm{d}G_k(\phi)(t)}{\mathrm{d}t} = f_k(\phi(t))$$

an application of Lemma 4.13, gives

$$\frac{\mathrm{d}F_k(\phi)(t)}{\mathrm{d}t} = \begin{cases} f_k(\phi(t)) & F_k(\phi)(t) > 0 \\ \max(0, f_k(\phi(t))) & F_k(\phi)(t) = 0 \end{cases}$$

Together with $F(\phi)(0) = \phi(0)$, this shows that x satisfies the differential equation if and only if $x = Fx$. The final step of the proof is to show that such a fixed point exists and that it is unique.

Consider two functions ϕ and ψ in V.

$$\begin{aligned} |G_k(\phi)(t) - G_k(\psi)(t)| &= \left| \int_0^t \big(f_k(s, \phi(s)) - f_k(s, \psi(s)) \big) \, \mathrm{d}s \right| \\ &\leq \int_0^t \big| f_k(s, \phi(s)) - f_k(s, \psi(s)) \big| \, \mathrm{d}s \\ &\leq \int_0^t C \max_\ell |\phi_\ell(s) - \psi_\ell(s)| \, \mathrm{d}s \\ &\leq C\, t \, \|\phi - \psi\| \end{aligned}$$

Since $C\, t \leq C\, T = 1/3$, it follows that

$$\|G(\phi) - G(\psi)\| \leq \frac{1}{3} \|\phi - \psi\|$$

and by Lemma 4.12, part (iii)

$$\|F(\phi) - F(\psi)\| = \|P(G(\phi)) - P(G(\psi))\| \leq 2\|G(\phi) - G(\psi)\| \leq \frac{2}{3} \|\phi - \psi\|$$

This proves that F is a contraction. By the Banach fixed point theorem, there exists a unique fix-point; a function $x \in V$ such that $Fx = x$. □

4.6 Stability for some linear time-delay system

If $\alpha > 0$ and $\tau > 0$, then the scalar system

$$\dot{x}(t) = -\alpha x(t - \tau)$$

is asymptotically stable if and only of $\alpha\tau < \pi/2$. In this section, we derive sufficient stability conditions for a couple of related systems.

Scalar system with multiple delays

Theorem 4.17 *If $\alpha_k > 0$, $\tau_k > 0$ and $\sum_k \alpha_k \tau_k \leq 1$, then the delayed differential equation*

$$\dot{x}(t) = -\sum_k \alpha_k x(t - \tau_k)$$

is globally asymptotically stable.

Proof: The characteristic function is $f(s) = s + \sum \alpha_k e^{-s\tau_k}$, which is an entire function. The condition $\sum_k \alpha_k \tau_k \leq 1$ implies that

$$\operatorname{Im} f(i\omega) = w - \sum \alpha_k \sin \tau_k \omega > w - \sum \alpha_k \tau_k \omega \geq 0$$

for all $w > 0$. Furthermore, $f(0) > 0$. An application of the argument principle, using the standard contour around the right half plane, proves that $f(s)$ has all zeros in the left half plane. □

Two-dimensional systems

We first consider the case where both states are subject to a delay,

$$\dot{x}(t) = -Ax(t-1) \qquad\qquad A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

The corresponding system, without delay, is stable if and only if both the trace and the determinant are positive. Furthermore, if there is no cross coupling, $a_{12} = 0$ or $a_{21} = 0$, the system is stable provided that the diagonal elements lie in the interval $0 < a_{11}, a_{22} < \pi/2$.

Theorem 4.18 *Let $D = \det A = a_{11}a_{22} - a_{12}a_{21}$ and $R = \operatorname{tr} A = a_{11} + a_{22}$, and assume that*

$$\begin{aligned} 0 &< R, D \\ D &\leq R/2 \\ D + \frac{\pi}{2} R &\leq \frac{\pi^2}{4} \end{aligned}$$

Then the system

$$\dot{x}(t) = -Ax(t-1)$$

is globally asymptotically stable.

Proof: Form the characteristic function,

$$f(s) = \det(sI + Ae^{-s}) = s^2 + Rse^{-s} + De^{-2s}$$

Apply the principle of variation to the characteristic function of this delayed differential equation, using the standard contour around the right half-plane. We have

$$\begin{aligned} f(i\omega) = -\omega^2 &+ R\omega \sin \omega + D \cos 2\omega \\ &+ i\omega \cos \omega \left(R - 2D \frac{\sin \omega}{\omega} \right) \end{aligned}$$

Since $R \geq 2D$, $\operatorname{Im} f(i\omega) > 0$ for all $0 < \omega < \pi/2$. Furthermore, for all $\omega > 0$, we have

$$\operatorname{Re} f(i\omega) < -\omega^2 + R\omega + D$$

The right hand side is a quadratic, with maximum at $R/2 < \pi/2$. Hence, for all $\omega \geq \pi/2$, we have

$$\operatorname{Re} f(i\omega) < -\frac{\pi^2}{4} + \frac{\pi}{2}R + D \leq 0$$

We see that for any $\omega > 0$ at least one of the inequalities $\operatorname{Im} f(i\omega) > 0$ or $\operatorname{Re} f(i\omega) < 0$ holds. This means that the curve avoids the fourth quadrant. Since we also have $f(0) = D > 0$, it follows that the curve does not encircle, nor pass through, the origin. By the argument variation principle, $f(s)$ does not have any roots in the right half-plane, and hence the system is stable. □

Corollary 4.19 *Assume that*

$$\begin{aligned} 0 < R, D \\ \tau^2 D \leq \tau R/2 \\ \tau^2 D + \frac{\tau\pi}{2} R \leq \frac{\pi^2}{4} \end{aligned}$$

Then the system

$$\dot{x}(t) = -Ax(t-\tau)$$

is stable.

Proof: Make the change of variable $s = t/\tau$. Then

$$\frac{\mathrm{d}x}{\mathrm{d}s}(s) = -\tau A x(s-1)$$

and we only need to note that $\det \tau A = \tau^2 \det A$ and $\operatorname{tr} \tau A = \tau \operatorname{tr} A$. □

Next, we consider a system with a particular structure which we will need in Chapter 6.

Theorem 4.20 *Let A, B, C and D be positive, and assume that*

$$\begin{aligned} A \leq 1 \\ BD \leq C \leq D \leq 1 \end{aligned}$$

Then the system

$$\dot{x}(t) = -\begin{pmatrix} A & 0 \\ A & 0 \end{pmatrix} x(t) - \begin{pmatrix} 0 & B \\ -D & B+C \end{pmatrix} x(t-1)$$

is globally asymptotically stable.

Proof: Form the characteristic function

$$f(s) = s^2 + As + \big(AC + (B+C)s\big)e^{-s} + BDe^{-2s}$$

We use the standard contour around the right half-plane. Put $s = i\omega$, then

$$\begin{aligned} f(i\omega) &= -\omega^2 + (B+C)\omega\sin\omega + AC\cos\omega + BD\cos 2\omega \\ &\quad + i\omega\left\{A + (B+C)\cos\omega - AC\frac{\sin\omega}{\omega} - BD\frac{\sin 2\omega}{\omega}\right\} \end{aligned}$$

Consider the interval $0 < \omega \le \pi/2$. Then

$$\begin{aligned} \frac{\operatorname{Im} f(i\omega)}{\omega} &= A\left(1 - C\frac{\sin\omega}{\omega}\right) + (B+C)\cos\omega - 2BD\cos\omega\frac{\sin\omega}{\omega} \\ &> (C + B - 2BD)\cos\omega \ge B(1-D)\cos\omega \ge 0 \end{aligned}$$

We also have $f(0) = AC + BD > 0$. Next, consider the real part, and $\omega \ge \pi/2$. First, assume that $\pi/2 \le \omega \le \pi$. Then $\cos\omega \le 0$, and $\sin\omega \ge 0$, so that

$$\begin{aligned} \operatorname{Re} f(i\omega) &= -\omega^2 + (B+C)\omega\sin\omega + AC\cos\omega + BD\cos 2\omega \\ &\le -\omega^2 + (B+D)\omega\sin\omega + BD\cos 2\omega \end{aligned}$$

Now, fix an ω in the interval. We have two cases: If $\omega\sin\omega \ge \cos 2\omega$, then the largest value is attained when $B = D = 1$, and

$$\operatorname{Re} f(i\omega) \le -\omega^2 + 2\omega\sin\omega + \cos 2\omega = (\cos\omega - \sin\omega + \omega)(\cos\omega + \sin\omega - \omega) < 0$$

For the final inequality, the second factor is clearly negative, and the first factor is positive, since $\cos\omega - \sin\omega + \omega \ge -\sqrt{2} + \pi/2 > 0$. Otherwise, if $\omega\sin\omega < \cos 2\omega$, then the maximum value is attained in the limit as $B = 1$ and $D \to 0$, or vice versa. Hence

$$\operatorname{Re} f(i\omega) \le -\omega^2 + \omega\sin\omega = \omega(\sin\omega - \omega) < 0$$

This shows that $\operatorname{Re} f(i\omega) < 0$ for all $\pi/2 \le \omega < \pi$. Finally, assume that $\omega \ge \pi$. Then

$$\begin{aligned} \operatorname{Re} f(i\omega) &= -\omega^2 + (B+C)\omega\sin\omega + AC\cos\omega + BD\cos 2\omega \\ &\le -\omega^2 + 2\omega + 2 = -(\omega-1)^2 + 3 \\ &\le -(\pi-1)^2 + 3 < 0 \end{aligned}$$

Since $f(0) = AC + BD > 0$, and $f(i\omega)$ avoids the fourth quadrant for all $\omega > 0$, it follows that the curve does not encircle, nor pass through, the origin. By the argument variation principle, $f(s)$ does not have any roots in the right half-plane, and hence the system is stable. □

4.7 Summary

We have investigated the stability properties of the ACK-clock mechanism, which is the inner-loop in window-based congestion control. Using the joint link model from the previous chapter, stability has been rigorously proven in three different settings: We have global asymptotic stability for the single link topology with a single flow and arbitrary delays, we have local asymptotic stability for the single link topology with multiple flows and arbitrary heterogeneous delays, and we have global asymptotic stability for an arbitrary topology but with no signalling delays.

For the single link, single flow topology, we also give bounds for the convergence time, and we see that convergence time depends not only on the propagation delay and the equilibrium queue delay, but also on the amount of cross-traffic.

Beyond the application to window-based congestion control, we have also stated and proved two general theorems on the existence and uniqueness of solutions to differential equations with state constraints, with or without delays. Finally, we have derived sufficient conditions for the stability of certain types of linear time-delayed systems of low dimension.

Chapter 5

Analysis of inter-protocol fairness

When evaluating proposed TCP improvements, one important issue is that of fairness, and in particular fairness in a mixed environment where the old and new TCP versions coexist and share the same resources. This chapter studies the inter-protocol fairness between TCP Westwood+ and TCP New Reno. As TCP Westwood+ can reduce the congestion window less than TCP New Reno, one might suspect that TCP Westwood+ takes a larger share of the available resources. On the other hand, it is known that TCP New Reno under-utilizes the bandwidth in the case of small buffer sizes. So does TCP Westwood+ just take this unused part of the bandwidth?

This chapter is organized as follows. Sec. 5.1 gives an introduction to the problem and related work. The system model is described in Sec. 5.2. Sec. 5.3 derives our analytical throughput results. In Sec. 5.4 discusses the results, and presents `ns2` simulations. We also propose a simple modification to Westwood+, to improve fairness over networks with large buffers.

5.1 Introduction

In [87], the impact of the buffer size on the performance of TCP Westwood+ was studied, and it was shown that unlike TCP New Reno, Westwood+ can achieve full link utilization with arbitrarily small link buffers. This makes Westwood+ an interesting alternative, not only for the wireless networks for which it was originally designed, but also for networks with large bandwidth delay product, where TCP New Reno needs large router buffers in order to achieve full link utilization. Since the performance of New Reno and Westwood+ depends on the buffer size in very different ways, one can expect that the buffer size at the bottleneck will be one of the most important parameters in the analysis.

The problem of the choice of the buffer size for regular TCP traffic has drawn significant attention [11, 9, 38, 51, 12, 48]. Here we show that the router buffer size has a significant influence not only on the efficiency of the network but also on the

fairness between different TCP versions. In particular, we show that in order for TCP Westwood+ to take a fair share of the available bandwidth, one needs to choose the buffer size of the bottleneck router not smaller than half of the bandwidth-delay product.

The performance of TCP Westwood+ over channels with loss and delay processes that are independent was studied in [7]. For the current fairness study, packet losses are due to congestion, and hence we can no longer assume that the loss process is independent of the window sizes.

We provide a model that describes the interaction of flows using TCP Westwood+ and TCP New Reno, with the bottleneck router buffer. Since queueing delays and packet losses due to congestion are coupled to the queue and window sizes, it becomes necessary to take queue and buffer size into account. This is in contrast to the paper [7], which analyzes TCP Westwood+ over channels where the loss and delay processes are independent. The evolution of the window sizes of both flows is modelled as a hybrid system [55], with a continuous increase between congestion events, and discontinuous reductions at the congestion event. This model lets us calculate analytically the throughput for the flows for any buffer size. The results are also validated using `ns2` simulations.

Both the analytical model and simulations give the same results: If the buffer size equals the bandwidth-delay product, TCP Westwood+ and TCP New Reno will share available capacity equally. For smaller buffers, Westwood+ wins the battle for capacity, and for larger buffers, TCP New Reno wins. Furthermore, we show that the results are only sensitive to a single parameter: the ratio between the buffer size and the bandwidth delay product. Finally, we propose a simple modification to TCP Westwood+ that makes TCP Westwood+ efficient even in the case of large buffers.

5.2 System model

We use a hybrid fluid flow model for a network with two competing TCP sources: TCP New Reno and TCP Westwood+. They share a single bottleneck router with a drop-tail buffer of size b and with transmission capacity c. The state variables of interest are the queue size $q(t)$ at the bottleneck router and the congestion window sizes $w_i(t), i = 1, 2$, of each TCP source. We assign index 1 to the TCP New Reno flow and index 2 to the TCP Westwood+ flow. The variable $q(t)$ is regarded as continuous and $w_i(t)$ as piece-wise continuous. The average sending rate of each source is one full window of data per round trip time, where the round trip time consists of a constant propagation delay τ_i and a queueing delay $q(t)/c$ that is determined by the current queue size. The window size corresponds to the amount of data that a source has transmitted, but not yet received any acknowledgment (ACK) for.

For the window dynamics, we focus on the congestion avoidance mode of TCP. As described in Sec. 2.2, congestion avoidance in TCP New Reno is based on additive increase / multiplicative decrease: The window is increased by one packet per round

trip time as long as ACKs are received. When a packet loss is detected, the window size is set to one half of the value before the loss.

In TCP Westwood+ [86], the additive increase is the same. The difference is that the multiplicative decrease is replaced by a different mechanism based on estimation of the available bandwidth and of the propagation delay. The original motivation was to improve TCP performance over wireless links, with a significant rate of losses that are due to transmission errors rather than congestion. When Westwood+ detects a packet loss, it sets its window size to the product of the estimated bandwidth before the loss, and the end-to-end propagation delay. The idea is that this window size is sufficiently small to allow queues on the path to drain, but not smaller.

The propagation delay estimate is simple: it is just the minimum observed round trip time. As long as at least one packet has been sent when the queue is close to empty, this estimate is accurate. The bandwidth estimation is more complex. We have to refer to the Westwood literature for the details [86], but the main idea is to obtain "bandwidth samples" from the stream of ACKs, and form the estimate by low-pass filtering of these samples. The first proposed version of Westwood formed a bandwidth sample for each ACK, while Westwood+ collects a round trip time worth of ACKs before forming a bandwidth sample. The estimation process and the resulting bias is analyzed in [61].

For the sending rate, we use the joint model for window-based transmission (Chapter 3), the sending rate for congestion control schemes such as TCP New Reno and TCP Westwood+ is

$$r_i(t) = \frac{w_i(t)}{\tau_i + q(t)/c} + \frac{\mathrm{d}w_i(t)}{\mathrm{d}t} \tag{5.1}$$

Here, we use the simplified joint link model, without no signalling delays, Eq. 3.7. The evolution of the window sizes is governed by the additive increase of TCP's congestion avoidance mode, which is the same for both New Reno and Westwood+. The window is increased by one packet, m_i bytes, each RTT.

$$\frac{\mathrm{d}w_i(t)}{\mathrm{d}t} = \frac{m_i}{\tau_i + q(t)/c}. \tag{5.2}$$

The queue size is related to the sending rates by $\mathrm{d}q(t)/\mathrm{d}t = \sum r_i(t) - c$, or, substituting (5.1) and (5.2)

$$\frac{\mathrm{d}q(t)}{\mathrm{d}t} = \sum_i \frac{w_i(t) + m_i}{\tau_i + q(t)/c} - c. \tag{5.3}$$

The above equation ignores signaling delays, but it includes the dependence on the queue size, via the RTT $\tau_i + q(t)/c$. Since the queue size must be non-negative, (5.3) is valid only when $q(t) > 0$ or the right hand side is positive; otherwise $\mathrm{d}q/\mathrm{d}t = 0$.

At the congestion event, the flow or flows that lose packets make a discontinuous change in its window size. For TCP New Reno, $w_1(t^* + 0) = w_1(t^* - 0)/2$. For

TCP Westwood+, $w_2(t^* + 0) = \text{RTT}_{\min}\hat{c}_2$, where $\hat{c}_2$ is the estimate of the flow's bandwidth, and $\text{RTT}_{\min}$ is the smallest observed RTT.

For TCP Westwood+, there are several issues with how this should be modeled.

- $\text{RTT}_{\min}$: This is intended to be the round trip delay, excluding queueing delay. So it makes sense to put $\text{RTT}_{\min} = \tau$. But on the other hand, if the Westwood+ flow is started when the bottleneck is already loaded, Westwood cannot observe τ, and it may be more accurate to set $\text{RTT}_{\min} = \tau + \min q(t)/c$.
- $\hat{c}_2$: The bandwidth is "sampled" once per round trip time, and then these samples are low-pass filtered to form a smoother estimate. The simplest model is to put $\hat{c}_2 = r_2(t^* - 0) = w_2(t^* - 0)/(\tau + b/c)$. This is an assumption of optimistic estimation, and it neglects the delay and the bias which are present in the real filter.

With $\text{RTT}_{\min} = \tau$ and $\hat{c}_2 = w_2(t^* - 0)/(\tau + b/c)$, it follows that $w_2(t^* + 0) = \beta w_2(t^* - 0)$, with the constant $\beta = c\tau/(c\tau + b)$.

5.3 Analysis

To analyze the system evolution, we use separate models for the evolution between congestion events, and for the congestion events. A *congestion event* is a short period of time when the router queue is full, and one or more packets are dropped. Finally, we put these two models together to find the stationary behavior, and the corresponding throughput.

System evolution between congestion events

Assume that a congestion event ends at time $t = 0$. At this time $q(0) = b$, i.e., the buffer is full, and the window sizes are given by the initial conditions $w_i(0) = w_i^0$.

After a congestion event, the evolution of the rates and the queue can be divided into three phases.

Phase 1: During the first phase, the total rate, $r_1 + r_2$, is smaller than c, and increasing. In this phase, the queue is shrinking, and it may even become empty.

Phase 2: During the second phase (which is present only for small buffer sizes), the queue is empty, and sending rates are increasing. During the second phase, the link is under-utilized, and the phase ends when the total sending rate reaches the link capacity, $r_1 + r_2 = c$.

Phase 3: During the third phase, the ACK-clock mechanism forces the total sending rate to stay essentially constant, slightly larger than capacity, $r_1 + r_2 = c + (m_1 + m_2)/(\tau + q/c)$. The growing windows result in a growing queue, not increasing sending rates. The third phase is terminated by the next congestion event, which happens when the queue size reaches the buffer size, $q(t) = b$.

The objective of the analysis in this subsection is to find the smallest $t^* > 0$ such that $q(t^*) = b$, and to express t^* and the corresponding window sizes $w_i(t^*)$ as functions of the initial window sizes w_i^0. These functions are needed in Sec. 5.3, when deriving the evolution over a large number of congestion events.

When investigating fairness between two flows, using $\tau_1 \neq \tau_2$ would introduce a prejudice, favoring one of the flows against the other. To give the two flow equal opportunities, we assume $\tau_1 = \tau_2 = \tau$. This assumption lets us introduce a virtual time, which can be thought of as measuring time in number of round trips. This tool makes it possible to find explicit solutions to the differential equations.

Virtual time s is defined by $\mathrm{d}t = (\tau + q(t)/c)\,\mathrm{d}s$. This change of variables transforms Equations (5.3) and (5.2) into

$$\frac{\mathrm{d}w_i(s)}{\mathrm{d}s} = m_i \tag{5.4}$$

$$\frac{\mathrm{d}q(s)}{\mathrm{d}s} = -q(s) - c\tau + \sum_i (w_i(s) + m_i) \tag{5.5}$$

The second equation is still valid only when $q(s) > 0$ or the right hand side is positive. Consider initial conditions $w_i(0) = w_i^0$, and $q(0) = q_0$. Introduce the notation $W = w_1^0 + w_2^0$ and $M = m_1 + m_2$. Then the solution can be written as

$$w_i(s) = w_i^0 + sm_i \tag{5.6}$$

$$q(s) = q_0 e^{-s} + (W - c\tau)(1 - e^{-s}) + Ms \tag{5.7}$$

Looking more closely at this solution $q(s)$, it can be divided into a transient, related to the initial buffer q_0 and a new "equilibrium size" $W - c\tau$, and a linear growth with rate M. Asymptotically, for large s, we have $q(s) \approx W + Ms - c\tau$, i.e., the queue is the difference between the total window size and the bandwidth-delay product $c\tau$.

Let s^* denote the virtual time corresponding to t^*, i.e., the next congestion event. Given s^*, the amount of data that is transmitted up to time t^*, D_i, can be computed as follows. First solve (5.4), which gives,

$$\begin{aligned} D_i &= \int_0^{t^*} r(t)\,\mathrm{d}t = \int_0^{t^*} \frac{w_i(t) + m_i}{\tau + q(t)/c}\,\mathrm{d}t \\ &= \int_0^{s^*} (w_i^0 + (s+1)m_i)\,\mathrm{d}s = s^* \left(w_i^0 + m_i + \frac{m_i s^*}{2} \right) \end{aligned} \tag{5.8}$$

This says roughly that the amount of data is the number of round trips, s^*, times the average window size. Here, the "average" is not a proper time average, but an average with respect to the virtual time s. We will compute the throughput as D_i/t^*. The calculation of s^* and t^* depends on the buffer size.

In general, t^* can computed given $q(s)$ and s^*, by integrating

$$t^* = \int_0^{t^*} \mathrm{d}t = \int_0^{s^*} \frac{\mathrm{d}t}{\mathrm{d}s}\,\mathrm{d}s = Ts^* + \frac{1}{c}\int_0^{s^*} q(s)\,\mathrm{d}s$$

The calculations in absolute time, t, are simplified by the remarkable fact that (5.2) and (5.3) can be solved for q in terms of w. Substitution of (5.2) into (5.3) gives

$$\begin{aligned}\frac{\mathrm{d}q(t)}{\mathrm{d}t} &= -c + \sum_i \frac{w_i(t)+m_i}{m_i} \cdot \frac{\mathrm{d}w_i(t)}{\mathrm{d}t} \\ &= \frac{\mathrm{d}}{\mathrm{d}t}\left\{-ct + \sum_i \frac{1}{2m_i}(w_i(t)+m_i)^2\right\}\end{aligned}$$

Integrating, we get

$$q(t_2) - q(t_1) = -c(t_2 - t_1) + \frac{1}{2}\sum_i \frac{(w_i(t_2)+m_i)^2 - (w_i(t_1)+m_1)^2}{m_i}$$

For any time interval, during which the queue stays non-empty, this allows us to compute the length of the interval given only the initial and final state,

$$t_2 - t_1 = \frac{1}{c}\left\{q(t_1) - q(t_2) + \frac{1}{2}\sum_i \frac{(w_i(t_2)+m_i)^2 - (w_i(t_1)+m_i)^2}{m_i}\right\} \tag{5.9}$$

The expressions for the time interval, $t_2 - t_1$, and for the amount of data, D_i, will be used to compute the throughput as $D_i/(t_2 - t_1)$.

Full utilization The link will be fully utilized if and only if $q(t) > 0$ for all time (except possible for an isolated instant). In other words, the second phase is empty. This section derives conditions on the initial conditions for this to happen.

Substitution of the initial condition $q_0 = b$ into (5.7) gives

$$q(s) = be^{-s} + (W - c\tau)(1 - e^{-s}) + Ms \tag{5.10}$$

This equation is valid only as long as $q > 0$, since the differential equation does not model queue underflow.

Proposition 5.1 (Link utilization) *Let $c > 0$, $\tau > 0$ and $b > 0$ denote the link capacity, the propagation delay, and the buffer size, respectively. Consider the queue evolution just after a congestion event at time $s = 0$, with $q(0) = b$. Let W be the sum of the flows' window sizes just after a congestion event, $W = w_1^0 + w_2^0$, and let $M = m_1 + m_2$. There are three possible cases, depending on the parameters and the initial condition, which can be characterized as follows.*

(i) If $W + M \geq c\tau + b$, then $q(s) > b$ for all $s > 0$. Such initial conditions violate the assumption that the previous congestion event ended at $s = 0$.

(ii) Otherwise, the link is fully utilized if and only if one of the following inequalities hold

$$W + M \geq c\tau \tag{5.11}$$

$$\frac{b}{M} \geq \exp\left(\frac{c\tau - W - M}{M}\right) - \frac{c\tau - W}{M} \tag{5.12}$$

Proof: Differentiation of Eq. (5.10) gives

$$\frac{\mathrm{d}q(s)}{\mathrm{d}s} = (W - c\tau - b)e^{-s} + M = (W + M - c\tau - b)e^{-s} + M\left(1 - e^{-s}\right)$$

If $W + M \geq c\tau + b$, then this expression is positive for all $s > 0$, and hence $q(s)$ is strictly increasing. This proves (i).

So assume that $W + M < c\tau + b$. Then we see that $\dot{q}(0) < 0$, and $q(s)$ has a unique minimum at

$$s_0 = \log \frac{c\tau + b - W}{M} > 0$$

We have full utilization if and only this minimum value $q(s_0)$ is non-negative. By substituting s_0 into (5.10), we see that $q(s_0) \geq 0$ is equivalent to Eq. (5.12). To see that also (5.11) implies full utilization, rewrite Eq. (5.10) as

$$q(s) = be^{-s} + (W + M - c\tau)(1 - e^{-s}) + M\left(e^{-s} + s - 1\right)$$

Since $e^{-s} + s - 1 \geq 0$ for all s, we see that $W + M - c\tau \geq 0$ implies $q(s) > 0$.

□

Large buffer If the buffer is "large", i.e., the condition in Prop. 5.1 is satisfied, then the queue never underflows, and we can find s^* by putting $q(s) = b$ in (5.10). The solution can be expressed in terms of Lambert's function, described in Sec. B. We use the two real branches, denoted W_0 and W_{-1}.

Proposition 5.2 *Assume that the full-utilization condition of Prop. 5.1 is satisfied, and use the same notation. Assume that $q(0) = b$. Define*

$$t^* = \min_t \{t > 0 | q(t) = b\}$$

and let s^* be the corresponding event in virtual time s. Then the values of s^* and t^* are given by

$$s^* = \tilde{s} + W_0\left(-\tilde{s}e^{-\tilde{s}}\right) \tag{5.13}$$

$$t^* = \frac{s^*}{c}\left(W + M + \frac{Ms^*}{2}\right) \tag{5.14}$$

where $\tilde{s}$ is defined by

$$\tilde{s} = \frac{c\tau + b - W}{M}$$

Proof: Put $q(s) = b$ in (5.10). After some simplifications, this equality implies

$$(s - \tilde{s})e^{s-\tilde{s}} = -\tilde{s}e^{-\tilde{s}}$$

This equation has two real solutions, the trivial one $s = 0$, and a second solution which can be expressed using the W_0 function, resulting in (5.13). With this value for s^*, (5.14) follows from (5.9). □

The throughput can be computed from (5.8),

$$\frac{D_i}{t^*} = c\,\frac{w_i^0 + m_i + m_i s^*/2}{W + M + M s^*/2}$$

As expected, with full utilization, the total throughput, $(D_1 + D_2)/t^*$, equals c. If the first flow has larger initial window size than the second, $w_1^0 > w_2^0$, and the same or larger packet size, $m_1 \geq m_2$, then the first flow gets a larger share of the capacity.

Small buffers When computing the throughput for a small buffer, we must handle the three phases separately, since neither the differential equation for q, nor the time interval equation (5.9), is valid during the second phase, when the queue is empty and the link is under-utilized.

Proposition 5.3 *When the full-utilization condition of Prop. 5.1 is* not *satisfied, the values of s^* and t^* are given by*

$$s^* = \frac{c\tau + b - W}{M} = \tilde{s} - 1 \tag{5.15}$$

$$t^* = \tau\,(\tilde{s} - 1) + \frac{b^2}{2cM} + \frac{s_1}{c}\left(W + M - c\tau + \frac{s_1}{2}M\right) \tag{5.16}$$

where s_1 is defined by

$$s_1 = \frac{c\tau - W}{M} + W_{-1}\left(-\frac{c\tau + b - W}{M}\exp\left(-\frac{c\tau - W}{M}\right)\right)$$

Proof: Denote the duration of the three phases, in absolute time and virtual time, by t_1, t_2, t_3, s_1, s_2, and s_3. We handle one phase at a time.

The first phase: We find s_1 by putting $q(s_1) = 0$ in (5.10) and solving for s_1. We get

$$\left(s_1 - \frac{c\tau - W}{M}\right)\exp\left(s_1 - \frac{c\tau - W}{M}\right) = -\frac{c\tau + b - W}{M}\exp\left(-\frac{c\tau - W}{M}\right)$$

By assumption, the full utilization conditions are not satisfied. Since (5.11) does not hold, we have $c\tau - W > 0$, and the right hand side is negative. Since (5.12) does not hold, the right hand side is larger than $-1/e$. The equation then has two real solutions, from the two real branches W_0 and W_{-1}. The one of interest to us is the smallest one, which gives

$$s_1 = \frac{c\tau - W}{M} + W_{-1}\left(-\frac{c\tau + b - W}{M}\exp\left(-\frac{c\tau - W}{M}\right)\right)$$

The time-interval equation, (5.9), gives

$$t_1 = \frac{1}{c}\left(b + s_1(W + M + M\frac{s_1}{2})\right)$$

The second phase: Throughout this phase, $q(0) = 0$. The phase starts with $\sum w_i(s_1) = W + s_1 M < c\tau - M$, and ends when $\sum w_i(s_1 + s_2) = W + (s_1 + s\_2)M = c\tau - M$. It follows that

$$s_2 = \frac{c\tau - W - M}{M} - s_1$$
$$t_2 = \tau s_2 = -\tau - \tau W_{-1}\left(-\frac{c\tau + b - W}{M}\exp\left(-\frac{c\tau - W}{M}\right)\right)$$

The third phase: We now solve the queue dynamics with the initial conditions $\sum w_i(s_1 + s_2) = c\tau - M$ and $q(s_1 + s_2) = 0$. With these initial condition, the solution (5.7) reduces to

$$q(s_1 + s_2 + s) = Ms$$

The phase ends when $q(s_1 + s_2 + s) = b$. This gives $s_3 = b/M$. The time-interval equation, (5.9), gives

$$t_3 = \frac{b\tau}{M} + \frac{b^2}{2cM} - \frac{b}{c}$$

Finally, addition of the values for each phase gives

$$s^* = s_1 + s_2 + s_3 = \frac{c\tau + b - W - M}{M} = \tilde{s} - 1$$
$$t^* = \tau\frac{c\tau + b - W - M}{M} + \frac{b^2}{2cM} + \frac{s_1}{c}\left(W + M - c\tau + \frac{s_1}{2}M\right)$$

□

Throughput Propositions 5.1–5.3 let us compute t^* and s^* as functions of the initial window sizes. With these values, the data volume D_i follows from (5.8), and the throughput D_i/t^* for each flow can be computed as

$$\frac{D_i}{t^*} = \frac{s^*(w_i^0 + m_i + m_i s^*/2)}{t^*}$$

For reference in the next section, we define the functions $\tilde{s}(W)$ and $s^*(W)$ as follows:

$$\tilde{s}(W) = \frac{c\tau + b - W}{M} \tag{5.17}$$
$$s^*(W) = \begin{cases} \tilde{s} + W_0\left(-\tilde{s}e^{-\tilde{s}}\right) & \text{full utilization} \\ \tilde{s} - 1 & \text{otherwise} \end{cases} \tag{5.18}$$

System evolution at congestion events

The previous section analyzed the queue evolution between congestion events. To find the average throughput over a longer time, we need to know the stationary

behavior that results from a large number of congestion events. We first consider what happens at the single congestion event at time $t = t^*$.

A congestion event means one or more packet is lost. As described in Sec. 5.2, New Reno's window update at a packet loss is $w_1(t^* + 0) = 0.5w_1(t^* - 0)$, and Westwood's window update at a packet loss is $w_2(t^* + 0) = \beta w_2(t^* - 0)$, where $\beta = c\tau/(c\tau + b)$.

Then, an important issue is *which* flow loses packets at a congestion event. It could be one of the flows, or both. In the TCP fairness literature, it is common to use stochastic modeling for this (see e.g., [14, 76]).

Under the fairly general assumption that the probabilities of the different possible outcomes at a congestion event depends only on the window sizes of the involved flows just prior to the congestion, the evolution can be described as a Markov chain. Let X^k denote the vector of the two window sizes just *after* a congestion event. If the first flow uses TCP New Reno and the second uses TCP Westwood+, the evolution can be described as

$$X^{k+1} = L^k G(X^k) \tag{5.19}$$

In this equation

$$G\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} + s^*(w_1 + w_2)\begin{pmatrix} m_1 \\ m_2 \end{pmatrix}$$

represents the window growth between congestion events, and the function $s^*(W)$ is defined by (5.18). The actual packet loss is represented by the random variable matrix L^k. Each L^k takes one out of three possible values,

$$L_{\text{Both}} = \begin{pmatrix} 1/2 & 0 \\ 0 & \beta \end{pmatrix} \qquad L_{\text{Reno}} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1 \end{pmatrix} \qquad L_{\text{Westwood}} = \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix}$$

Of the three outcomes, L_{Both} represents a loss event where each flow loses a packet, L_{Reno} represents an event where only the first flow loses a packet, and L_{Westwood} represents an event where only the second flow loses a packet.

If we make the further assumption that the probability distribution of L^k depends only on the state X^k, i.e.,

$$\mathrm{P}(L^k = L|X^k, X^{k-1}, \ldots, X^0, L^{k-1}, L^{k-2}, \ldots, L^0) = \mathrm{P}(L^k = L|X^k)$$

then (5.19) clearly describes a Markov chain.

To gain insight into the problem, and to be able to solve it analytically, we have to make a few simplifying assumptions and approximations. Our first assumption is that flows are fully synchronized, i.e., that at every congestion event, both flows lose packets. This means that $L^k = L_{\text{Both}}$ for all k, and it actually makes the process fully deterministic.

The second simplification is dropping the non-linear terms of the function $s^*(w_1, w_2)$. The $W_0(\cdots)$-term in Eq. (5.18) is bounded; it always lies between -1 and 0. So by replacing s^* with $\tilde{s} - 1/2$, the error in $G(X^k)$ is at most half a packet.

To give both flows equal opportunities, we also put $m_1 = m_2 = m$. The result is the following approximation of the evolution of the state X^k.

$$\begin{aligned} X^{k+1} &= L_{\text{Both}}\left(X^k + m\left(\tilde{s}(X_1^k, X_2^k) - 1/2\right)\begin{pmatrix}1\\1\end{pmatrix}\right) \\ &= \begin{pmatrix}1/4 & -1/4\\ -\beta/2 & \beta/2\end{pmatrix} X^k + (c\tau + b - m)\begin{pmatrix}1/4\\ \beta/2\end{pmatrix} \end{aligned} \tag{5.20}$$

Proposition 5.4 *Let the capacity be c, the propagation delay τ, packet size m, and b the size of the router buffer. Define $\beta = c\tau/(c\tau + b)$. Then, for arbitrary initial values X^0, the iteration of Eq. 5.20 converges to limit value X^*, given by*

$$X^* = \frac{c\tau + b - m}{3 - 2\beta}\begin{pmatrix}1-\beta\\ \beta\end{pmatrix} \tag{5.21}$$

Proof: Since $\beta < 1$, the matrix has all eigenvalues within the unit circle, which implies that the iteration converges. The limit value can be computed as

$$\begin{aligned} X^* &= \left(I - \begin{pmatrix}1/4 & -1/4\\ -\beta/2 & \beta/2\end{pmatrix}\right)^{-1}(c\tau + b - m)\begin{pmatrix}1/4\\ \beta/2\end{pmatrix} \\ &= \frac{c\tau + b - m}{3 - 2\beta}\begin{pmatrix}1-\beta\\ \beta\end{pmatrix} \end{aligned}$$

□

To find the throughput for both flows in the stationary regime, we plug in the window sizes X^* as the initial window sizes in the procedure of Sec. 5.3.

Limits

Let us study the fairness in the limiting cases when buffers are very small or very large. Of particular interest is the throughput ratio

$$\frac{D_2}{D_1} = \frac{2w_2^0 + m(s^* + 1)}{2w_1^0 + m(s^* + 1)}$$

Proposition 5.5 *In the limit as $b \to \infty$, Westwood+ gets one quarter of the capacity, while New Reno gets three quarters. In the limit as $b \to 0$ and $m \to 0$, Westwood+ gets all the capacity, and New Reno gets 0.*

Proof: For the large buffer case, note that $c\tau + b = c\tau/\beta$. We parameterize the expressions in terms of β, and let $\beta \to 0$. We have from (5.21) that

$$\begin{pmatrix}w_1^0\\ w_2^0\end{pmatrix} = X^* = \frac{1}{3}\begin{pmatrix}c\tau/\beta + O(1)\\ c\tau + O(\beta)\end{pmatrix}$$

and $W = c\tau/(3\beta) + O(1)$. It is clear that we have full utilization, and we get

$$s^* = \frac{c\tau}{3m\beta} + O(1)$$

Finally,

$$\frac{D_2}{D_1} = \frac{2w_2^0 + m(s^*+1)}{2w_1^0 + m(s^*+1)} = \frac{1/3 + O(\beta)}{2/3 + 1/3 + O(\beta)} \to \frac{1}{3}$$

For the small buffer case, simply letting $b \to 0$ would give $W \to c\tau - m$. Then $W + M - c\tau - b \to -m < 0$, which violates part (i) of Prop. 5.1. The smallest possible b can be found by solving

$$b^2 + \left(c\tau + \frac{5m}{2}\right) b - \frac{mc\tau}{2}$$

So in general, both b and the ration D_2/D_1 depends on the relation between m and $c\tau$. Here, we handle only the usual case that $m \ll c\tau$. We put $b = m$, and let $m \to 0$. Then

$$W + M = c\tau + \frac{6m^2}{c\tau + 3m}$$

and it follows that $W + M > c\tau$, so that we have full utilization, and that $W + M < c\tau + b = c\tau + m$ for all $m < c\tau/3$. We also have the limits

$$\tilde{s} = 1 + \frac{1}{2}\,\frac{c\tau - 3m}{c\tau + 3m} \to 3/2$$
$$s^* = \tilde{s} + W_0(-\tilde{s}e^{-\tilde{s}}) \to 0.874$$

Furthermore, Eq. (5.21) also implies that $w_1^0 \to 0$ and $w_2^0 \to c\tau$. It follows that

$$\frac{D_2}{D_1} = \frac{2w_2^0 + m(s^*+1)}{2w_1^0 + m(s^*+1)}$$

grows to ∞ as $m \to 0$. □

This result is different from what one could obtain by using the throughput formulas for the average throughput with TCP New Reno, T^{Reno}, and the average throughput with Westwood+, T^{Westwood}, as given in [52]:

$$T^{\text{Reno}} = \frac{1}{\text{RTT}}\sqrt{\frac{2(1-p)}{p}}, \qquad T^{\text{Westwood}} = \sqrt{\frac{1-p}{p\tau_q \text{RTT}}}.$$

In these equations, p is the average loss probability, τ_q is the average queueing delay, and $\text{RTT} = \tau + \tau_q$ is the average round-trip time. Let us form the throughput ratio:

$$\frac{T^{\text{Westwood}}}{T^{\text{Reno}}} = \frac{1}{\sqrt{2}}\sqrt{1 + \frac{\tau}{\tau_q}}$$

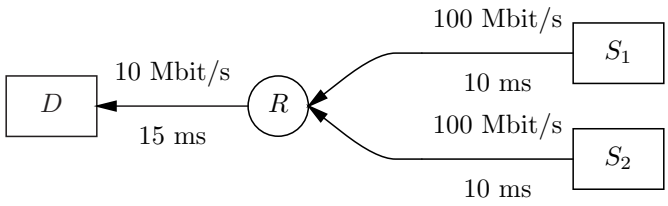

Figure 5.1: Experimental setup. Parameter values correspond to scenario #1.

We see that this ratio predicts fairness when $\tau_q = \tau$, i.e., when the *average* queuing delay equals the propagation delay. This is slightly different from our model, which predicts fairness when the *maximum* queuing delay, b/c, equals the propagation delay.

In the small buffer limit, $\tau_q \to 0$, the ratio tends to infinity. So here we have perfect agreement. In the large buffer limit, $\tau_q \to \infty$, the ratio tends to $1/\sqrt{2}$, significantly higher than the prediction 1/3 from our analysis. The simulation results presented in the next section will support our analysis in predicting the trend of the throughput ratio for large buffers. Our explanation for this discrepancy is in the random loss assumption made by the square root formula, which does not hold in our setting.

5.4 Results and discussion

Theoretical results

We use the network topology illustrated in Fig. 5.1. There are two source nodes, S_1 (New Reno) and S_2 (Westwood+), sending data to a single destination node, D. There is one intermediate router, R, and the link between the router and the destination is the network's bottleneck. For the bottleneck link, we use two values for the capacity c, 1 Mbit/s and 10 Mbit/s, and two different values of the propagation delay, corresponding to round trip propagation delay τ of 50 and 200 ms. Table 5.1 shows the parameter values for the four scenarios, and the corresponding bandwidth-delay product. Both sources use a packet size of 1500 bytes. For all scenarios, we vary the buffer size of the router between 2 packets and roughly twice the bandwidth-delay product, $2c\tau$.

The resulting throughputs according to the analysis of Sec. 5.3, for scenario #1, is shown in Fig. 5.2. The solid line represents TCP New Reno, and the dashed line represents TCP Westwood+. The topmost dotted curve is the sum of the normalized throughputs, i.e, the link utilization. In [87] it was shown that Westwood+ achieves almost full utilization for arbitrary small buffer sizes, so it is not surprising that TCP Westwood+ dominates over New Reno for small buffer sizes. When the buffer

Scenario #	c [Mbit/s]	τ [ms]	$c\tau$ [packets]
0	1	50	4.2
1	10	50	42
2	1	200	16.7
3	10	200	167

Table 5.1: Network parameters for four different scenarios

size equals the bandwidth-delay product, both flows get the same throughput. This is expected from the model, since for this buffer size, $\beta = 1/2$, both flows use the same decrease factor after a loss, and both elements of X^* are equal.

For larger buffer sizes, Westwood+ suffers from its estimation of $\text{RTT}_{\min}$. It uses the smallest observed RTT, 50 ms, even though in stationarity, the buffer never gets empty, and the actual RTT stays significantly higher than Westwood's $\text{RTT}_{\min}$ at all times. This forces Westwood+ to reduce its window size, after a packet loss, even more than New Reno does.

In Fig. 5.3, the corresponding curves for all four scenarios are shown in the same figure. To aid the comparison, the horizontal axis has been scaled so that for each scenario, 1 corresponds to the bandwidth-delay product. The theoretical throughput curves for all four scenarios are plotted on top of each other, the increasing curve represents New Reno, the decreasing curve represents Westwood+, and the uppermost curve close to one is the utilization. The curves for the four scenarios are barely distinguishable. Scenario #3, with the largest bandwidth-delay product, is shown with solid curves, and for this we see that the link is slightly under-utilized for buffer sizes significantly smaller than the bandwidth-delay product. We can conclude that the single variable defined by $b/(c\tau)$ determines the bandwidth sharing between TCP Westwood+ and TCP New Reno.

Simulation results and validation

To validate the present model and the assumptions (such as fluid flow approach and full synchronization), we simulate the same scenarios using `ns2`, and compare these results to the theoretically computed values. We assign random start times to the two flows, measure the actual throughput for a long transmission, excluding transients at flow startup, and average over several realizations. The results, with 95% confidence intervals, are shown in Fig. 5.4–5.7. The vertical axis corresponds to the normalized throughput and the horizontal axis corresponds to the buffer size in packets. The `ns2` simulations are displayed for New Reno with marks $\times$, Westwood+ with marks $+$, and their sum with marks $\circ$. The curves show the theoretical results.

The `ns2` simulation and the analysis give throughput values that match remarkably well, except for very small buffer sizes ($b < 5$). The most likely cause of the mismatch for small values of b is the fluid model approach.

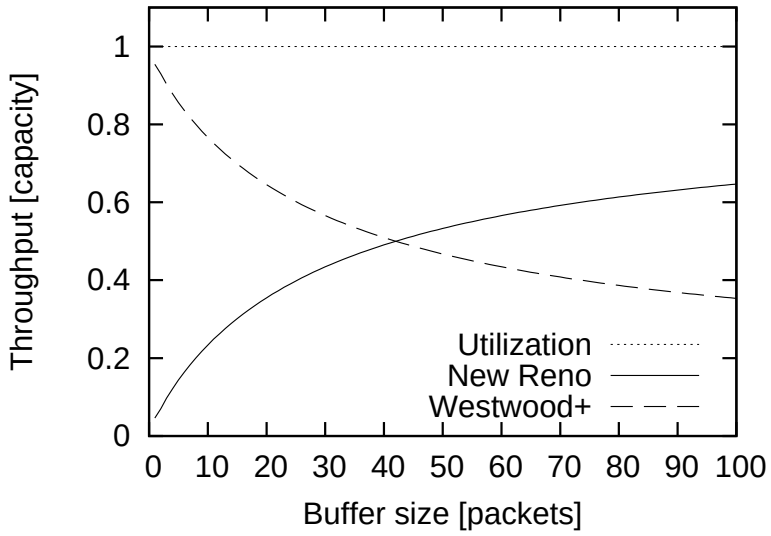

Figure 5.2: Throughput (normalized so that 1.0 is the link capacity) as a function of buffer size, for scenario #1.

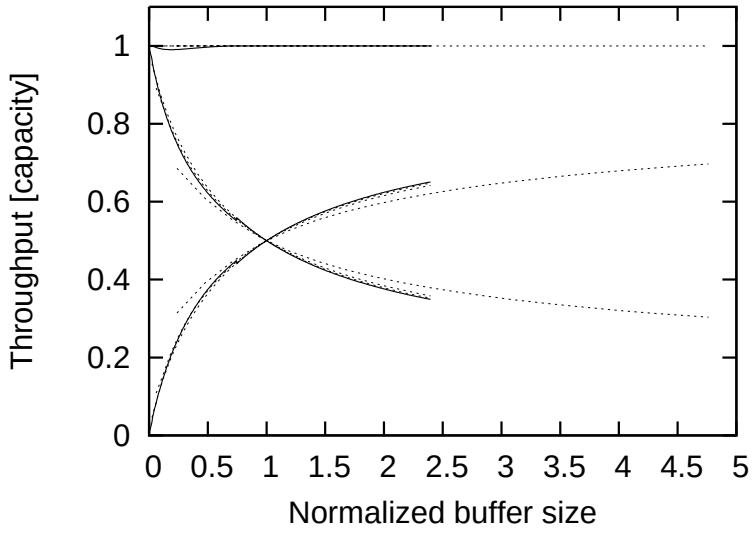

Figure 5.3: Normalized throughput as a function of normalized buffer size, $b/(c\tau)$. The theoretical throughput curves for all four scenarios are plotted together. The solid curves corresponds to scenario #3, with the largest bandwidth-delay product.

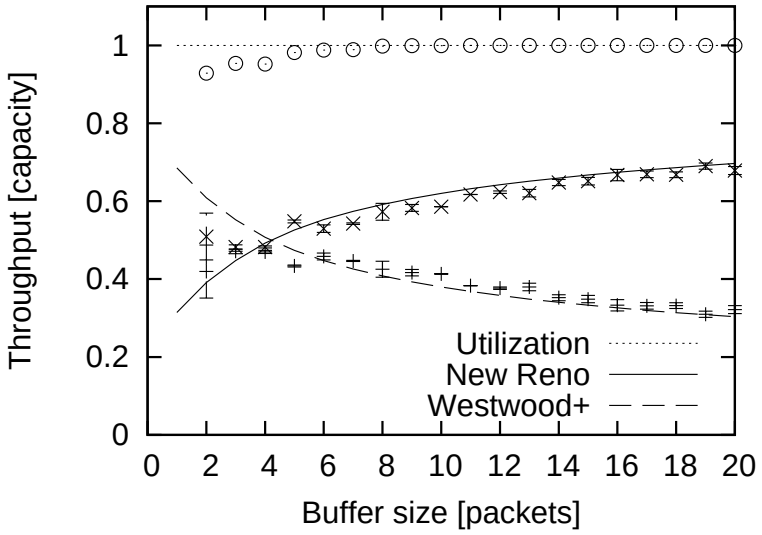

Figure 5.4: Validation for scenario #0. Throughput as a function of buffer size.

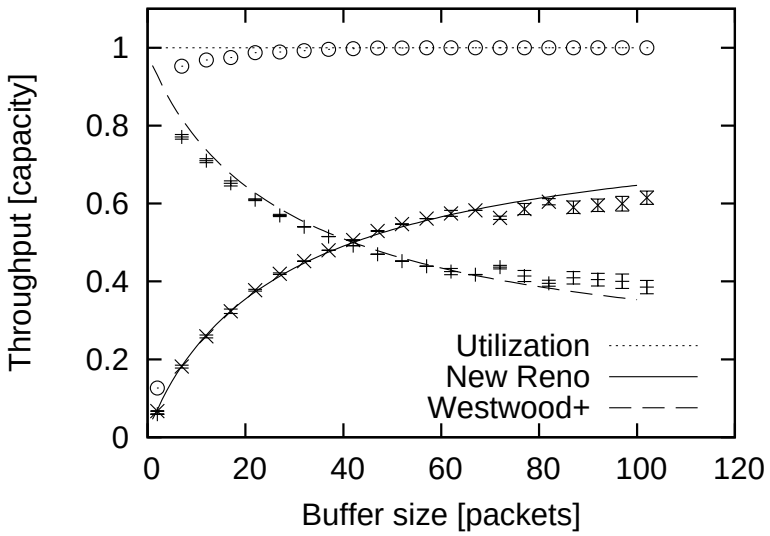

Figure 5.5: Validation for scenario #1. Throughput as a function of buffer size.

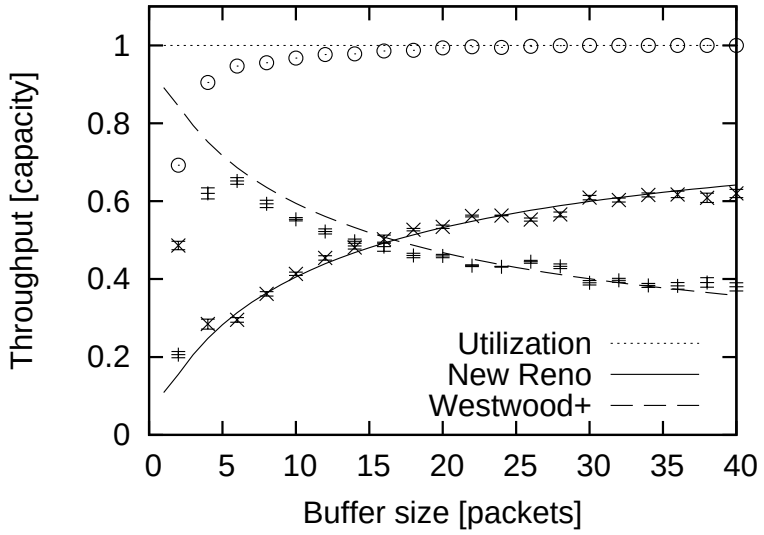

Figure 5.6: Validation for scenario #2. Throughput as a function of buffer size.

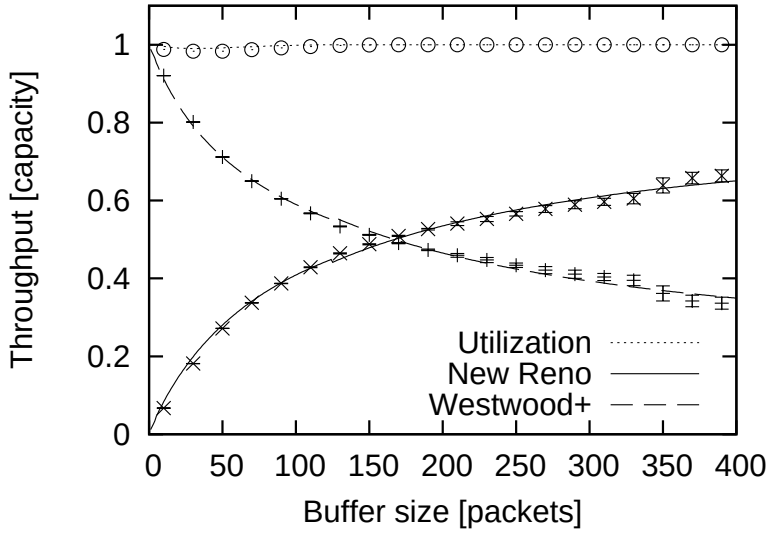

Figure 5.7: Validation for scenario #3. Throughput as a function of buffer size.

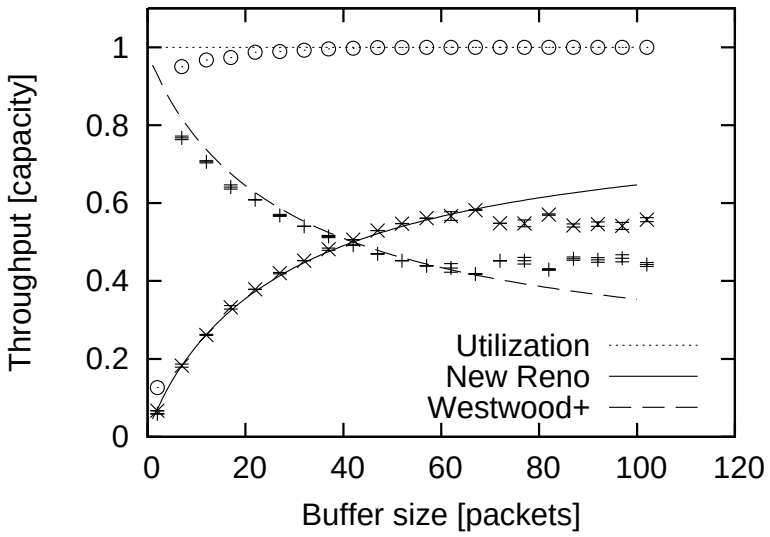

Figure 5.8: This figure shows the result for scenario #1, (c.f., Fig. 5.5), but with the start of the Westwood+ flow delayed around 50 s.

Consider the ratio between Westwood+ and New Reno throughput. As can be seen in the figure, the ratio decreases as the buffer size is increased and it tends to $1/3$ rather than the $1/\sqrt{2}$ that one obtains by application of the "square root formulas" in [52].

For TCP Westwood+ to function properly, it needs to estimate RTT_{min}. It shares this requirement with delay based congestion control methods such as TCP Vegas and Fast TCP. If a Westwood+ flow is started at a time when the bottleneck link is highly loaded, the buffer may stay close to full for the whole duration of the flow. Then, in effect, part of the queueing delay will be erroneously accounted for as propagation delay, which results in an overestimation of the real bandwidth delay product and hence in a less dramatic reduction of the congestion window. In this case, the disadvantage that Westwood+ has when competing with New Reno over large buffer, is actually reduced. This is illustrated in Fig. 5.8, which shows the result of a simulation of scenario #1 (c.f., Fig. 5.5), with the only difference being that the start of the Westwood+ flow is delayed around 50 seconds so that the TCP New Reno flow gets enough time to fill the buffer. For the larger buffer sizes, Westwood's RTT_{min} estimate has values up to 80ms, to be compared to the true round trip propagation delay of 50ms.

Furthermore, when the number of flows over the bottleneck links is increased, it becomes less probable that the router buffer empties. Thus, also in this case, one can expect the RTT_{min} estimate to include some of the queueing delay. And

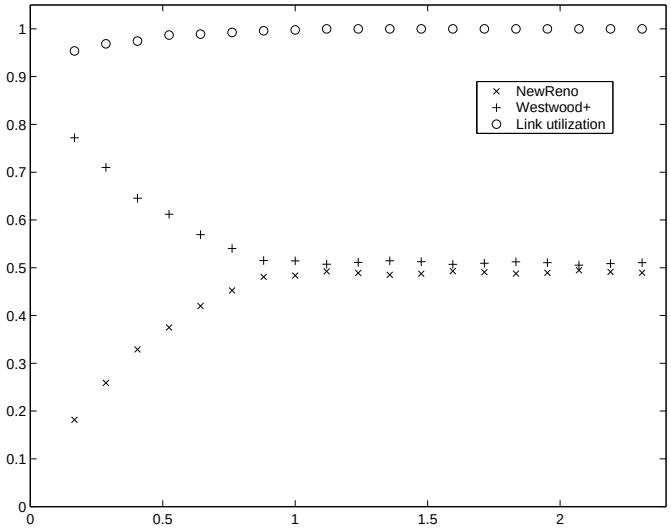

Figure 5.9: The performance of TCP Westwood+ in scenario #1 after our modification - Normalized throughput vs. normalized buffer size ($b/(c\tau)$).

consequently, the disadvantage of Westwood+ can be expected to be reduced when more flows are multiplexed.

Fairness with large buffers

To improve fairness in the large buffer case, the window decrease of Westwood+ in response to a packet loss could be modified, so that the new window is never set to a value smaller than half of the previous value. In other words, modify Westwood+ to never decrease its window more than New Reno would have done. To validate this claim, we modified the Westwood+ code in `ns2` accordingly and rerun the simulation for scenario #1. The results are presented in Fig. 5.9. Clearly, for buffers larger than the bandwidth delay product when it is very probable that TCP Westwood+ divides its window by more than 2, our modification solves the unfairness problem. At the same time, the performance of both protocols for buffers smaller than the bandwidth delay product stays the same.

Fairness with small buffers

For successful deployment of TCP Westwood+ as a general purpose congestion control mechanism over the Internet without significant negative effects on New Reno flows, it seems prudent not to set the size of buffers in routers to small values. It is known (see e.g., [11, 12]) that for small buffers, TCP New Reno under-utilizes the available bandwidth at the bottleneck link. In contrast, TCP Westwood+ is able

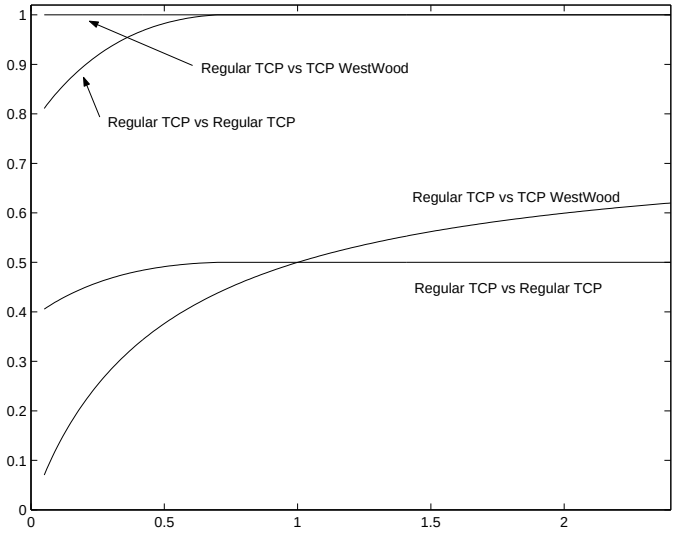

Figure 5.10: The normalized throughput of TCP New Reno when competing with either TCP New Reno or TCP Westwood+ as a function of the normalized buffer size.

to utilize all available bandwidth even when the buffers are small. Then one can ask the following question: Does TCP Westwood+ takes only unused bandwidth when it competes with TCP New Reno, or does it steal some bandwidth from New Reno?

To answer this question, we consider two TCP New Reno flows instead of the mix of TCP Westwood+ and TCP New Reno and we vary the ratio between the buffer size and the bandwidth delay product. The results are depicted in Fig. 5.10. In this figure, we plot normalized throughput of TCP New Reno when the competing TCP connection is either TCP New Reno or TCP Westwood+. The two topmost curves correspond to link utilization. We can read in this figure that when two TCP New Reno flows compete for the available bandwidth and the bottleneck buffer is small, the link is not fully utilized, as expected. The figure demonstrates that TCP New Reno does suffer from the presence of TCP Westwood+ flow when the buffer size is smaller than the bandwidth-delay product. For buffers larger than the bandwidth delay product, regular TCP realizes better performances after the introduction of TCP Westwood+.

5.5 Summary

In this chapter, we studied analytically and by the means of `ns2` simulations, the inter-protocol fairness between TCP Westwood+ and TCP New Reno. Until the present, the effect of the introduction of TCP Westwood+ on TCP New Reno was not

thoroughly investigated. We explained why these protocols when they compete for available bandwidth get different shares. Our main conclusion is that the bandwidth sharing only depends on the ratio between the buffer size at the bottleneck router and the bandwidth delay product. In particular, if the ratio is smaller than one, TCP Westwood+ takes more bandwidth. And if the ratio is greater than one, then it is TCP New Reno that gets the larger part.

The introduction of TCP Westwood+ allows to solve the well known problem of network under-utilization by regular TCP when buffer sizes in routers are set to small values. Unfortunately, this gain in the utilization comes at the expense of regular TCP which loses some of its throughput.

Inspired by our results, we proposed a simple modification to TCP Westwood+ that solves the unfairness problem for large buffer sizes. For small buffers, the unfairness problem is still open.

Chapter 6

A new congestion control protocol

CONGESTION control has been a very active area of research during the last decade. Many new algorithms have been proposed. Since TCP seems to work reasonably well in practice, most proposals are in the form of smaller modifications to TCP. Still, few of the proposals have seen wide-spread deployment on the Internet, and most of those that have been deployed, e.g., ECN [101], SACK [88, 18], and window scaling [60], are designed primarily using "common sense" arguments, rather than precise mathematical analysis.

This chapter develops a new congestion control mechanism. This mechanism provides for small and stable queues at bottlenecks, reasonable convergence times, and proportional fairness. This chapter is organized as follows. Sec. 6.1 reviews the control objectives, and the way standard TCP tries to achieve them. Our proposed protocol is described in Sec. 6.2, and Sec 6.3 discusses similarities and differences to related protocols.

Sec. 6.4 develops and analyzes a fluid flow model for the protocol, using a single link, single flow topology, and the joint link model to describe the inner-loop. We derive the equilibrium values of window size and queue size, and prove local stability. We get a very simple stability condition.

Sec 6.5 presents simulations of the fluid-flow model, and investigates the influence of the different parameters in the system and in the controller. In Sec. 6.6, we consider the equilibrium properties for an arbitrary topology, and see that the protocol is approximately proportional fair. The utility weights are similar to TCP's; a ratio between packet size and propagation delay.

In Sec 6.7, we present simulation results from an `ns2` implementation of the protocol, for topologies with one or several bottlenecks. Finally, we summarize our results and outline further work in Sec. 6.8.

6.1 Introduction

Requirements

Let us step back and review the control objectives for Internet congestion control, and how they are met by TCP. Our main focus is on the congestion avoidance phase of TCP, not the startup phase.

Avoid network overload: This is the most important objective, and it is achieved by TCP, as witnessed by the fact that the Internet has not suffered congestion collapse since the introduction of TCP congestion control in the late 1980s.

TCP controls the size of each flow, but it does not control the number of flows. There are extreme scenarios where the number of flows become very large, while the throughput of each flow becomes very small, less than a packet per RTT. For example, assume that a crowd of users are arriving with a rate of one user per second, and that each user tries to download a 2 Mbyte file from a web server. Then the traffic demand on that server is almost 17 Mbit/s. If the capacity is smaller, e.g., if there is a 10 Mbit/s bottleneck link connecting the server to the Internet, then the number of flows grow without bound, and so does the users' download time. In such a scenario, congestion control is not effective. Mechanisms that limit the number of flows are called *admission control*.

Efficient utilization: TCP usually achieves full utilization of bottleneck links, since the additive increase keeps growing the sending rate until some link on the path becomes a bottleneck. The problematic cases are short-lived flows and paths with very large bandwidth-delay product. There is also one somewhat hidden inefficiency: Congestion on backbone links would lead to large queueing delays and severe quality degradation, in particular for non-TCP real-time traffic. As a consequence, most backbone links are highly over-provisioned, or use complex quality-of-service mechanisms to protect non-TCP traffic. If TCP were friendlier to the bottlenecks, those resources could be spent elsewhere in the network.

Fair sharing: Competing TCP flows share available resources, with some favoring of flows with short RTT and large packet size.

So how does TCP implement fairness between flows? The additive increase means that in the absence of packet losses, each flow keeps increasing its sending rate. The increase depends on packet size and RTT, but it is independent of the flow size. Eventually, the additive increase ensures that some bottleneck will drop packets due to queue overflow.

Among the flows sharing a bottleneck, the packet loss probability is the same for all flows. This implies that the total number of packet losses are distributed in proportion to flow size; a flow of five packets per second will suffer on

average five times as many packet losses per unit time as a flow of only one packet per second. Each flow reacts to a packet loss by reducing its sending rate. Since the reduction is multiplicative, a large flow not only suffers a larger number of packet losses per unit time than a small flow, the reduction in sending rate will also be larger than for the small flow. As the cycle of additive increase and multiplicative decrease goes on, the average rates converge to the fair shares.

React to changes: The TCP steady state is that each flow keeps an average sending rate equal to the flow's fair share of the capacity (for some definition of "fair"). Changes to both network load and network topology can be interpreted as changes to this fair share. From this point of view, congestion control is a distributed estimation problem, where the objective is to estimate each flow's fair share.

Reaction to an increased capacity or fair share is limited by the additive increase mechanism, while the multiplicative decrease gives a fast response to the packet losses that result when capacity is decreased or traffic load is increased.

Small queueing delays: TCP does not attempt to minimize the queue size. Bottleneck routers will drop packets due to overflow of the router buffer. Furthermore, the buffer size is usually large, the old rule of thumb for buffer sizing corresponds to a queueing delay on the order of 200 ms when the buffer is close to full [26].

A simple approach to reduce queueing delay is to reduce the buffer size by one or a few orders of magnitude, compared to the old rule of thumb [9, 48]. Small buffers may however lead to lead to high packet loss rates and high variability in TCP throughput [38]. Another, more sophisticated, approach is to use AQM. The most widely implemented AQM scheme being Random Early Detection (RED). Good tuning of the parameters of AQM schemes, and RED in particular, is challenging [33, 56]. Both these approaches work best when there are a large number of TCP flows sharing the bottleneck, and they are less useful when there are only one or a couple of flows, e.g., in the scenario where two users or applications share an ADSL line.

Maintaining small queues does not necessarily mean that the system must be stable (i.e., that sending rates and queue sizes converge to equilibrium values). It is acceptable if queues exhibit oscillations or even chaotic behavior, as long as the oscillations are small.

For a new congestion control scheme to be deployable over the Internet, there are some additional constraints (see also Sec 1.3 for some more details) that must be taken into account.

Scalability: The algorithm must work for huge networks.

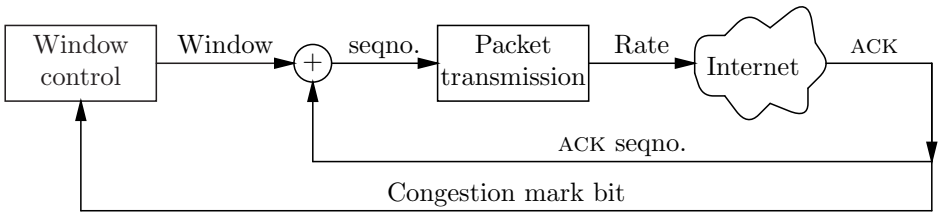

Figure 6.1: Control structure for the congestion control protocol.

End-to-end principle: As far as possible, protocol state should belong to the communication end points.

Robustness: Parameters such as number of flows, capacities, and delays vary over ranges of several orders of magnitude, and knowledge of the parameters is very limited. Hence, the algorithm must work despite severe uncertainty about the parameters.

Tunability: The number of tuning parameters should be small, and the effect of each parameter should be predictable and easy to understand.

Incremental deployability: A new algorithm must work in the setting that a subset of end-nodes and routers are upgraded, and flows using the old and the new algorithm share resources.

Openness: The protocols and mechanisms must be free to implement for anyone.

6.2 Protocol design

From the above objectives and constraints, we are looking for a new congestion control mechanism, which addresses the main shortcoming of TCP, i.e., the large queueing delays at bottlenecks. We intend to have no packet losses in normal operation, and instead rely on congestion feedback generated by the network. The proposed scheme is structurally very similar to standard TCP/AQM, with some subtle but crucial differences in the handling at both end hosts and routers. The control structure is illustrated in Fig. 6.1. In this section we describe and motivate the design choices.

Window-based control

The ACK-clock is a simple and efficient per-packet rule which controls the sending rate. Since this inner-loop is stable, for arbitrary propagation delay in the network (see Chapter 4), we keep the ACK-clock inner-loop, and just modify the window control. This gives the cascaded control structure in Fig. 6.1.

Additive increase

The lack of feedback in the absence of congestion, and the desire to have efficient utilization of resources, requires some mechanism that keeps the sending rates growing in the absence of congestion. This growth rate is going to be one of the system parameters, and for stability reasons, it makes sense to scale it down with the RTT. Hence, we also keep the additive increase part of TCP: in the absence of congestion, increase sending rates by m bytes per RTT, where m is a system parameter. Using only a linear increase seems prudent when there is some reason to expect that the system is close to congestion, or that the current sending rate is close the the fair share. (When we believe that the current rate is much smaller than our fair share, in particular when a flow is new, with a small initial window, a more aggressive increase rule may be used. However, the current analysis does not include any such slow-start like mechanisms.)

With the flow analogy in mind, additive increase can be thought of as a pressure applied by the sources, which forces more fluid into the system, filling up the pipes, and when the pipes are full, the pressure causes queues to grow.

So for an effective congestion control, we must design a way for the queues to impose a back-pressure on the fluid.

Additive decrease

With the multiplicative decrease mechanism of TCP New Reno, feedback is a relatively rare event, and the response to each feedback event is strong. If the average window size of a flow is n packets, varying between $2n/3$ and $4n/3$, then the loss probability per packet is $3/(2n^2)$ (see also the "square root formula" for TCP throughput, Eq. (2.1)), and the rate of losses per unit time is one loss per $2n/3$ RTTs.

Hence, the sender will operate for many RTTs between feedback events, growing its window according to the additive increase law. For drop tail queues, this is the reason large buffers are necessary for high utilization. The long interval between feedback events is a consequence of the combination of additive increase and multiplicative decrease. The average intervals are long also when using AQM and ECN in the routers.

We aim for more frequent signalling, on the average of one feedback event per RTT. To get there, we need a softer response to each event: For each received ACK carrying a congestion indication, the window size is decreased by one packet. To use a feedback signal that is added to the sender's congestion window is similar in spirit to XCP [63], although we use much lighter signalling. With this rule, senders will no longer operate in open-loop for long time periods.

Another important benefit is that from a router's point of view, the system becomes more predictable. Under the one-packet decrease rule, when a router decides to set a mark bit, the effect is that one packet less will be arriving a round-trip time later. For comparison, consider AQM with ECN and multiplicative decrease,

where the corresponding flow will halve its window. Since the window size is unknown, in this case both the timing and the *magnitude* of the effect is unknown. With the one-packet decrease rule, the magnitude of the response is known, only the round-trip time is uncertain.

Information contents The design of the marking rule is related to the research on control under information constraints. This field tries to link together fundamental control properties, such as the Bode sensitivity integral, information theoretic properties, such as channel capacity and mutual information, and estimation theory notions such as the Fischer information [85, 35].

What is the information contents in the feedback bits? Consider a mark bit y which is set by the network, with probability p for a one and $1-p$ for a zero. The probability p is a function of the network state that we would like to estimate. If p is modelled as a stochastic variable with probability density function $f_p(p)$, then the mutual information between y and p, denoted $I(y,p)$, is given by

$$\mathrm{I}(y,p) = h(\mathrm{E}[p]) - \mathrm{E}[h(p)]$$

where

$$h(p) = -(p \lg p + (1-p)\lg(1-p))$$

and lg denotes logarithm to base 2. The concavity of $h(p)$ implies $I(y,p) \geq 0$. (More details are provided in Appendix A). The amount of information thus depends on the distribution of p.

When the output y is used for feedback, p will depend on the network state which in turn depends on y. The author is not aware of any tools to compute the distribution of p in this setting. We can however get some insight by investigating a simpler estimation problem. Assume that p is uniformly distributed in the interval $[0, p_{\max}]$. The resulting mutual information, as a function of $p_{\max}$, is shown in Fig. 6.2. For small $p_{\max}$, we have $I(y,p) \approx 0.14\, p_{\max}$.

From this, one can get a crude estimate of the actual amount of feedback information. Consider a window w corresponding to $n = w/m$ packets, with some moderately large n, and assume an average marking probability of once per RTT, $\mathrm{E}[p] = 1/n$. Model the lack of information about p as a uniform distribution in $[0, 2/n]$. Then the mark bit in each received ACK provides approximately $0.28/n$ bits of information about p. In each RTT, we receive n ACKs, and hence we get 0.28 bits of information.

In contrast, consider the same average window size of n packets, and a marking probability $E[p] = 3/(2n^2)$, as for TCP New Reno with a network supporting AQM and ECN. Then we get only $0.42/n$ bits of information per RTT using the same calculation. To conclude, we see that the amount of feedback information available to the controller is higher with frequent feedback.

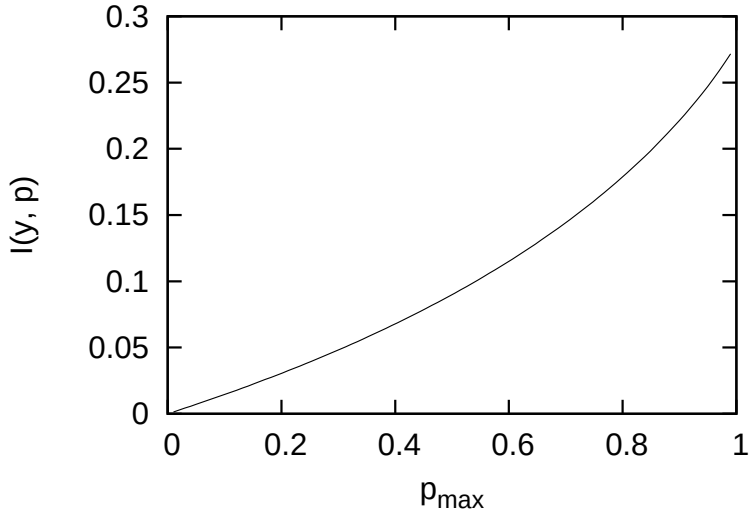

Figure 6.2: Mutual information for a uniformly distributed marking probability.

Congestion feedback

The amount of signalling that can be sent from the network to the sources is limited. One common approach is to piggyback a congestion indication on packets forwarded to the receiver, which copies the information into the corresponding acknowledgment and sends it back. One subtle advantage with this approach is that if the congestion signal is lost, that implies that a data packet (or acknowledgment) is lost too, which is going to be noticed by the sender. This is unlike the source quench messages that were used in the early days of TCP, and transmitted as independent ICMP (Internet Control Message Protocol) packets.

We will assume that we can allocate one bit of information in the packet header, analogous to the standardized ECN bits. The main difference to ECN is that the response we are defining for the sender is different, and also the rules for setting the bit are different.

The main reason to introduce this feedback is to get a signal that is proportional to flow size, and hence useful for fairness. Only bottleneck routers should add marks to forwarded packets, for several reasons

- The state of non-bottlenecks is generally not interesting enough to spend signalling resources on.
- There are usually a small number of bottlenecks along a given path. Combining the signalling from several routers is somewhat difficult both for implementation and for analysis, so the smaller the number, the better.

- For deployment, it is desirable that the new congestion control can be deployed by upgrading only bottleneck routers, and leave the large number of non-bottleneck routers unchanged.

One important consequence is that a network where no link is fully utilized will generate no packet marks. We thus get no quantitative information about how close the network is to full utilization.

Marking rule

It is important that the feedback rule uses a minimal number of parameters; one of the problems with current AQM is that the algorithms are difficult to tune so that they work well. We are going to consider rules with a single parameter.

For good control performance, delay is important. If the state is estimated using an explicit observer (at either sender or router), time constants must be kept small. For this reason, there is limited room for sophisticated estimation in the router. On the other hand, it is also impractical to base the feedback on instantaneous arrival rate; if a router receives a burst of a few packets back to back at a rate much higher than capacity, but the queue is empty before the arrival of this burst, that is no reason to signal to all sources using the queue that it has become congested.

As a compromise, we generate feedback from the instantaneous queue size. There are two reasons for this:

- The queueing delay is the primary value we want to control.

- The queue dynamics provides about the right amount of low-pass filtering of the arrival rate: The queue absorbs small fluctuations, and only if a disturbance is large enough to result in a significant change in the queue size, it is relevant for congestion control.

We will describe the feedback process as a stochastic procedure, where each packet is marked with some probability p that depends on the current queue size. The marking must not necessarily be implemented in a stochastic way, e.g., it may be possible to ensure that the right proportion of packets are marked, using virtual queues and some fully deterministic rules. In the latter case, the marking process will still appear to be stochastic when observed by sources, given that the cross-traffic is random.

The feedback to be attached to a packet can, in principle, be attached at any time the packet is in the queue. In practice, the most common alternatives are to mark packets either at arrival to the queue, or just before they are transmitted. In the analysis, we will assume that packets are marked at exit from the queue, since this minimizes the signalling delay. Marking packets at arrival has the advantage of giving a higher correlation between the marks seen by a flow and the burstiness of that particular flow.

We choose the simple one-parameter rule

$$p(q) = \frac{q}{q_0 + q}$$

where the q_0 parameter corresponds to the size of a severely congested queue, where 50% of the packets are marked, and hence q_0 can be much larger than the desired queue size. A large q_0 implies a small gain from q to p.

Another candidate was the function

$$p_{\text{alt}}(q) = \begin{cases} 0 & q \leq q_0 \\ \frac{q-q_0}{q} & q > q_0 \end{cases}$$

Since the equilibrium size at a bottleneck corresponds to a non-zero marking probability, the equilibrium queue size with the p_{alt} rule will always be larger than q_0. This means that q_0 must be of the same order as the desired queue size, i.e., small. From simulations with p_{alt} and a small q_0, it may happen that the system oscillates with peak queue size just above q_0, and then the high gain causes a significant decrease in the sending rate. The effect is that the system spends much of the time in the dead zone, in effect in open loop for long periods of time, just like TCP New Reno.

The problem with p_{alt} is not the dead-zone per se (looking close to an equilibrium, the dead zone has the same effect as a small additional propagation delay), but the high gain.

Incremental deployability

It is easy to make the sources fall back to TCP New Reno when some bottleneck on the path does not implement the new mechanisms. If we keep the rule that the window size is halved when a packet loss is detected, then we keep TCP New Reno's behavior both in the case that some bottleneck router on the path does not implement the packet marking scheme, and in the case of severe congestion.

To make routers handle a mix of older TCP versions and the new scheme is a little more difficult. One possibility is to use an additional bit in the packet header to distinguish between flows that implement the new scheme and flows that do not (analogous how the same issue is handled with ECN), and use separate queues for the two classes.

These deployment issues are not addressed in detail in this thesis.

6.3 Relation to other protocols

As already mentioned, the signalling structure is the same as in standard TCP/AQM. The new protocol is also related to delay-based congestion control, in particular TCP Vegas, in that small queue sizes is a design objective. However, delay feedback is used only in the inner-loop, i.e., the ACK-clock, while the outer-loop adjusts the

window size based on explicit feedback from the network. So in the cascaded control structure, the two loops use different feedback signals. Another important difference is that there is a per-link tuning knob. If a particular link needs a higher average queue size, e.g., due to a traffic mixed with larger fluctuations in the arrival rate, or a time varying capacity, the average queue size can be increased by local tuning.

A cascaded control structure is also used in [89]: The controller adjusts the window size, and then the corresponding sending rates and queue sizes are determined by the window sizes. The novelty in this work is that we take into account the dynamic properties of the relationship between window sizes and queue sizes, using the joint link model from Chapter 3.

The control structure is similar to that in primal-dual congestion control schemes, although window-based rather than rate-based. One difference is that there are two different signals that are fed back from the network to the sources: Delay, and a congestion indication. For this reason, the protocol does not fit easily into the utility maximization framework. This problem is shared with the family of loss-delay based congestion control.

We use a congestion indication, which, when it arrives to a source, is *added* to the source's congestion window. This is unlike the standard ECN mechanism, where sources are expected to treat the arrival of a congestion indication in the same way as a packet loss, and react by halving the window size. Additive feedback is used in the XCP protocol [63], but we use much lighter signalling.

6.4 Analysis

System model

For the stability analysis, we focus on the simplest topology, with a single source and a single bottleneck. Without loss of generality, we can assume that the forward delay, i.e., the time taken for a transmitted packet to arrive to the bottleneck queue, is zero. The queue dynamics, from Chapter 3, are

$$\dot{q}(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) + x(t) - c$$

This equation is valid only as long as $q(t) > 0$; if $q(t) = 0$ and the right hand side is negative, then the queue stays in the empty state and $dq(t) = 0$.

In the router, packets are marked with probability given by

$$p(q) = \frac{q}{q_0 + q}$$

The window update law uses additive update of one packet per RTT in the absence of marks. The packet size is denoted m. When marks are received, the window size is decreased with the corresponding amount. The expected number of received marks is np, where n is the window size in units of packets, $n = w/m$, and

p is the marking probability. For each received mark, the window size is reduced by m, so that the expected reduction during one RTT is wp. In the continuous time fluid flow model, this can be expressed as

$$\dot{w}(t) = \frac{m - p\big(q(t-\tau)\big)w(t)}{\tau + q(t-\tau)/c}$$

Put together, this gives the following system equations for the state variables $w(t)$ and $q(t)$:

$$\begin{aligned}
\dot{w}(t) &= \frac{1}{\tau + q(t-\tau)/c}\left(m - \frac{q(t-\tau)}{q_0 + q(t-\tau)}\, w(t)\right) \\
\dot{q}(t) &= \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \frac{1}{\tau + q(t-\tau)/c}\left(m - \frac{q(t-\tau)}{q_0 + q(t-\tau)}\, w(t)\right) + x(t) - c
\end{aligned} \tag{6.1}$$

Equilibrium

For constant cross-traffic, put $\gamma = (c-x)/c$. The equilibrium is the solution w^* and q^* of the equations

$$\begin{aligned}
mq_0 &= q^*(w^* - m) \\
w^* &= \gamma(q^* + c\tau)
\end{aligned} \tag{6.2}$$

For any positive parameters, a unique equilibrium exists, given by the intersection of the line $w(q) = \gamma c\tau + \gamma q$, and the curve $w(q) = m + mq_0/q$. It is clear that $q^* > 0$ and $w^* > \max(m, \gamma c\tau)$.

The equilibrium equations can be solved explicitly:

$$\begin{aligned}
w^* &= \frac{\gamma c\tau + m}{2} + \sqrt{\frac{(\gamma c\tau - m)^2}{4} + \gamma q_0 m} \\
q^* &= -\frac{\gamma c\tau - m}{2\gamma} + \frac{1}{\gamma}\sqrt{\frac{(\gamma c\tau - m)^2}{4} + \gamma q_0 m}
\end{aligned}$$

In particular, for large q_0, both q^* and w^* grow as $\sqrt{q_0}$. We can also solve for q_0 in terms of q^*,

$$q_0 = q^*\left(\frac{\gamma(c\tau + q^*)}{m} - 1\right)$$

To get an equilibrium queue of one packet, $q^* = m$, then requires that

$$q_0 = \gamma c\tau + \gamma m - m \le c\tau$$

So it should be a reasonable small-buffer tuning to put $q_0 = c\tau$; this choice gives $w^* = \gamma c\tau + m$ and $q^* = m/\gamma$.

Stability analysis

In this section, we study the system behavior close to the equilibrium.

Theorem 6.1 *Let $c > 0$ be the bottleneck capacity, $\tau > 0$ the propagation delay, and $m > 0$ the packet size. Assume that the control parameter q_0 is chosen so that $q_0 \geq c\tau$, and that the cross-traffic is constant and below capacity, $x(t) = x < c$. Then the system* (6.1)*, describing the queueing and window dynamics, is locally asymptotically stable.*

Proof: Define $\gamma = (c - x)/c$, and let q^* and w^* denote the equilibrium values, i.e., the solutions to Eq. (6.2). Put $q(t) = q^* + \widetilde{q}(t)$ and $w(t) = w^* + \widetilde{w}(t)$, and assume that $\widetilde{q} > -q^*$, so that the queue does not underflow. Ignoring higher order terms, the linearized system is

$$\dot{\widetilde{w}}(t) = \frac{1}{\tau + q^*/c}\left(-\frac{q^*}{q_0 + q^*}\widetilde{w}(t) - \frac{q_0 w^*}{(q_0 + q^*)^2}\widetilde{q}(t-\tau)\right)$$
$$\dot{\widetilde{q}}(t) = \frac{1}{\tau + q^*/c}\left(-\frac{q^*}{q_0 + q^*}\widetilde{w}(t) + \widetilde{w}(t-\tau) - \left(\gamma + \frac{q_0 w^*}{(q_0 + q^*)^2}\right)\widetilde{q}(t-\tau)\right)$$

Scale time, by the change of variables

$$v(t) = \begin{pmatrix} \widetilde{w}(\tau\, t) \\ \widetilde{q}(\tau\, t) \end{pmatrix}$$

Then the system is of the form

$$\dot{v}(t) = -\begin{pmatrix} A & 0 \\ A & 0 \end{pmatrix} v(t) - \begin{pmatrix} 0 & B \\ -D & B + C \end{pmatrix} v(t-1)$$

where

$$A = \frac{\tau}{\tau + q^*/c}\,\frac{q^*}{q_0 + q^*}$$
$$B = \frac{\tau}{\tau + q^*/c}\,\frac{q_0 w^*}{(q_0 + q^*)^2}$$
$$C = \gamma\frac{\tau}{\tau + q^*/c}$$
$$D = \frac{\tau}{\tau + q^*/c}$$

Since $w^* = \gamma c(\tau + q^*/c)$, the assumption $q_0 \geq c\tau$ implies that

$$B = \frac{\tau}{\tau + q^*/c}\,\frac{q_0 w^*}{(q_0 + q^*)^2} = \frac{q_0}{q_0 + q^*}\,\frac{\gamma c\tau}{q_0 + q^*} < \gamma$$

It is then clear that $0 < A < 1$ and $BD < C \leq D < 1$. Stability of the linearized system now follows from Theorem 4.20. Then the equilibrium of the non-linear is locally asymptotically stable. □

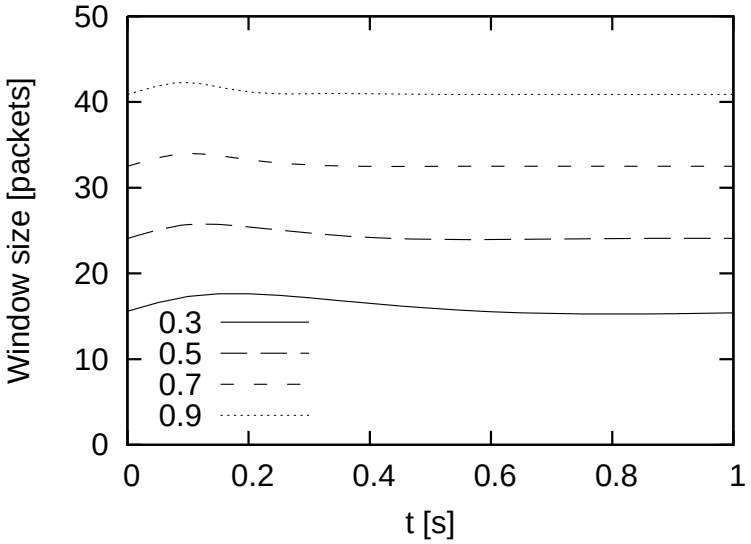

Figure 6.3: The evolution of the window size, for $\gamma = 0.3$, 0.5, 0.7, and 0.9.

6.5 Flow-level simulation

To get a better understanding of the dynamics, and to see the influence of the system parameters τ and γ, and the design parameter q_0, we perform a series of simulations based on the fluid-flow system model (6.1).

As base values, consider a 10 Mbit/s link, with a roundtrip propagation delay of 50 ms, and with 30% non-responsive cross-traffic (i.e. $\gamma = 0.7$). The packet size is 1500 bytes, and we set the control parameter q_0 to 150 packets. For comparison, the bandwidth delay product is 42 packets for the base delay, and 250 packets for a larger delay of 300 ms. We will then vary the parameters γ, τ and q_0 in turn. For all the simulations, the initial conditions are $q(t) = 0$ and $w(t) = w^*$ for $t \leq 0$.

We first vary γ. We have seen in Chapter 4.2 that the time constant of the inner-loop grow large when γ is decreased. In Figs. 6.3 and 6.4, we see that also the dynamics of the closed loop window control gets slower as γ is made smaller, i.e., when the amount of cross-traffic is increased. We can also see that when γ is decreased, the equilibrium queue grows, while the equilibrium window size shrinks.

Next, consider the influence of the propagation delay τ. In Figs. 6.5 and 6.6 we see the dynamics when the delay is increased from 50 ms up to 300 ms. By Theorem 6.1, the system is locally stable provided $\tau \leq q_0/c = 180$ ms. From the curves corresponding to a propagation delay of 50 ms and 100 ms, the system is stable with a convergence time of about 5 times the propagation delay. For 200 ms delay, the system is still stable, but with much slower convergence. Finally,

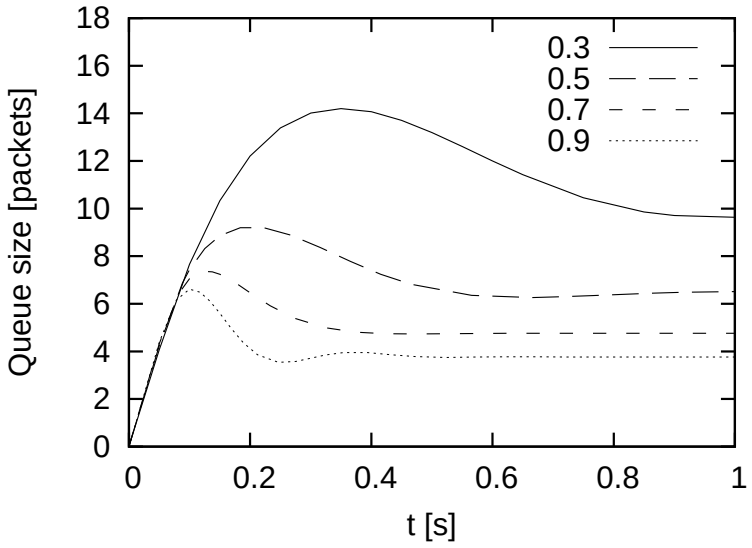

Figure 6.4: The evolution of the queue size, for $\gamma = 0.3$, 0.5, 0.7, and 0.9.

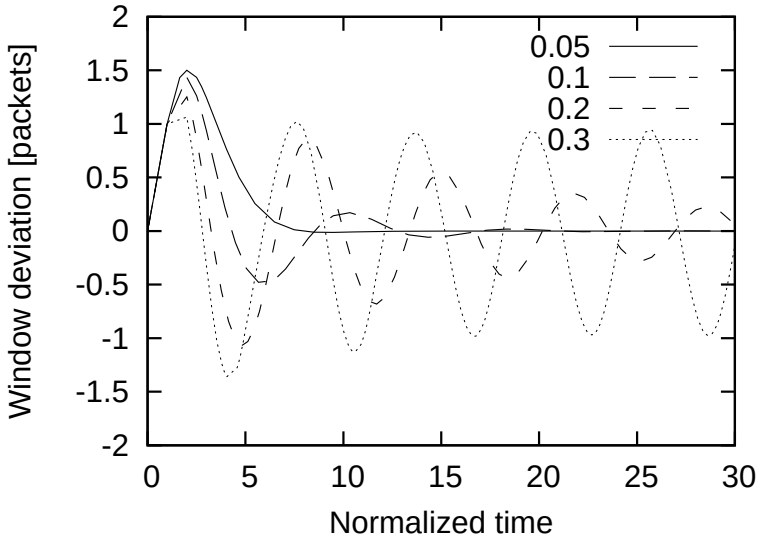

Figure 6.5: The evolution of the window size, for $\tau = 0.05$, 0.1, 0.2, and 0.3. To aid comparison, the vertical axis shows the deviation from the equilibrium (w^* increases with τ), and the horizontal axis is in units of τ.

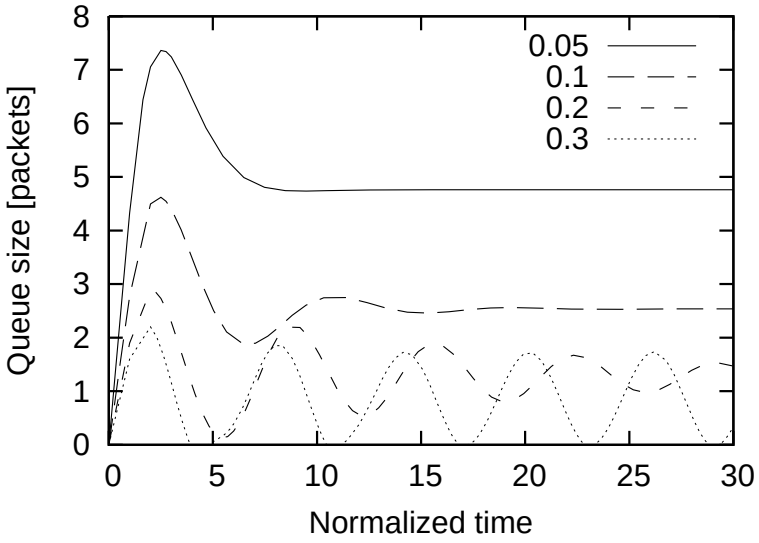

Figure 6.6: The evolution of the queue size, for $\tau = 0.05$, 0.1, 0.2, and 0.3. To aid comparison, the horizontal axis is in units of τ.

for 300 ms delay, the system approaches a limit cycle where the queue oscillates between zero and approximately two packets. Since the cycle includes an interval when the queue is empty, there is also a slight underutilization of the link in the scenario with 300 ms delay.

For all the values of τ considered here, stable or unstable, the size of transients and oscillations in the queue size are at most a couple of packets. For τ even larger than 300 ms, both the oscillations and the utilization would get smaller, and q_0 must be increased to recover full utilization.

In Figs. 6.7 and 6.8, we see the influence of the design parameter q_0. By Theorem 6.1, the system is stable when $q_0 \geq 42$ packets. We see that increasing q_0 makes the system less oscillatory, and the equilibrium values for the window size and the queue size larger. For $q_0 = 40$ packets, the system converges slowly. For the smallest value, $q_0 = 10$ packets, the system is unstable and converges to a limit cycle. The amplitude of this limit cycle is quite small, in comparison to the transients for stable choices of q_0. So from a practical perspective, the main problem caused by this instability is a waste of resources, since the queue gets empty which implies underutilization of the bottleneck link, not large delay and delay fluctuations degrading the quality of real-time applications.

Finally, we examine the dependence on the propagation delay τ, when the design parameter q_0 is scaled with τ. Figures 6.9 and 6.10 show the dynamics for a propagation delay τ from 100 ms up to 3 s, when $q_0 = c\tau$. The equilibrium window

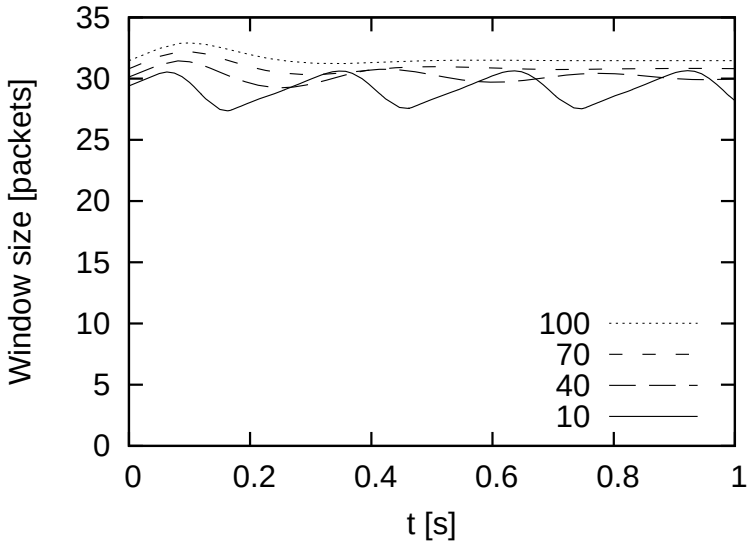

Figure 6.7: The evolution of the window size, for $q_0 = 10$, 40, 70, and 100.

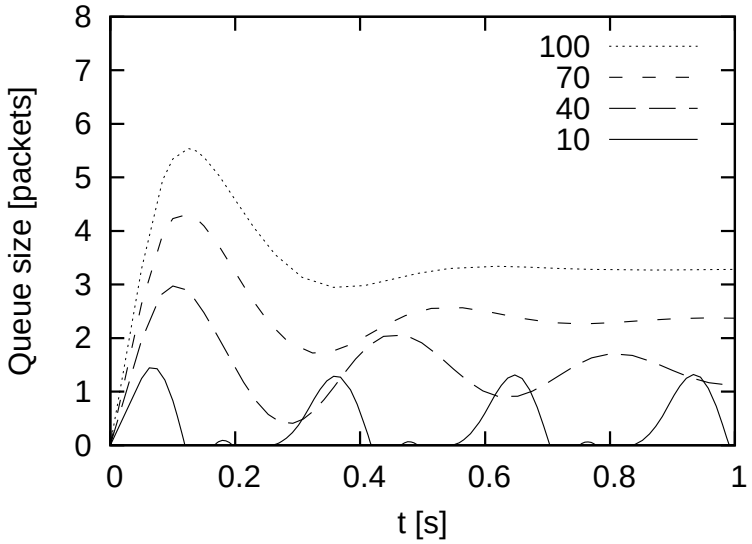

Figure 6.8: The evolution of the queue size, for $q_0 = 10$, 40, 70, and 100.

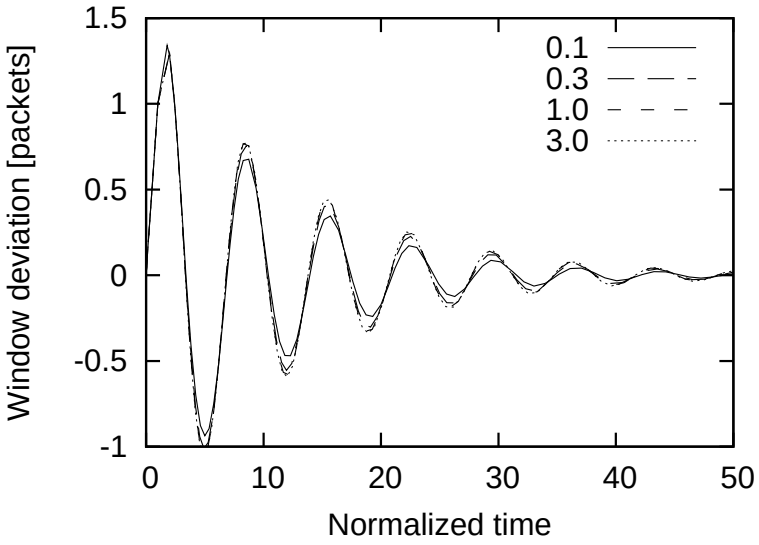

Figure 6.9: The evolution of the window size, for $\tau = 0.1$, 0.3, 1.0, and 3.0, and $q_0 = c\tau$. To aid comparison, the vertical axis shows the deviation from the equilibrium (w^* increases with τ), and the horizontal axis is in units of τ.

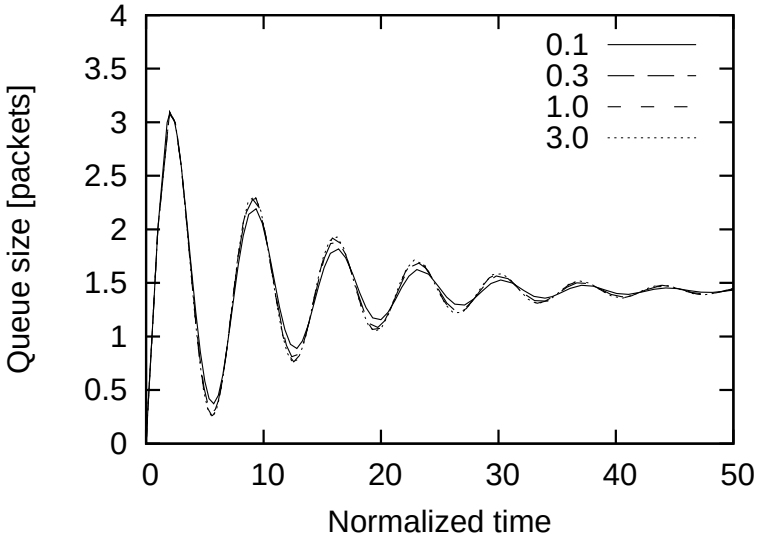

Figure 6.10: The evolution of the queue size, for $\tau = 0.1$, 0.3, 1.0, and 3.0, and $q_0 = c\tau$. To aid comparison, the horizontal axis is in units of τ.

size w^* grows with τ, but except for this, increasing τ corresponds only to a scaling of the time axis. The system is somewhat oscillatory. Increasing q_0 to $1.5c\tau$ or $2c\tau$ would reduce the oscillations. We also see that the convergence time is on the order of 20τ. For comparison, by Prop. 4.5, the convergence time of the inner-loop is bounded by 3.4τ.

Tuning

These simulations indicate that tuning q_0 is reasonably simple. A too small q_0 causes underutilization and moderate queue oscillations. A too large q_0 results in an unnecessarily large stationary queue, but since $q^* = O(\sqrt{q_0})$ for large q_0, q_0 can be a few times too large with no severe effects. So simple tuning rules, along the following lines, can be used:

- Set q_0 at each link to roughly the largest expected bandwidth-delay product of any flow using the link. (This procedure is very similar to the traditional rule-of-thumb of router buffer sizing).
- If oscillations that cause queue underflow are observed, increase q_0 by a factor of two.
- If the queue has a significant stationary queue, decrease q_0 by 30%.

Finally, note that the dynamics of the congestion control mechanisms is not the only consideration in the tuning of q_0. The queue must also absorb any stochastic fluctuations in the arrival rate, so if there are large fluctuations, a larger stationary queue is needed, and hence a larger q_0.

6.6 Fairness

In this section, we consider the equilibrium properties in a general network of links and sources using the proposed scheme. We use the notation from the utility maximization framework (Sec. 2.3). At each source, we have a window size w_s and a sending rate r_s. At each link, we have a queueing delay d_ℓ.

The network topology is described by a routing matrix A, and the roundtrip propagation delay τ_s^0 for each source. Sources are aggregated as $y = Ar$, and the delays are aggregated as $\tau = A^T d$.

Since we will be concerned with equilibrium properties, rather than the actual dynamics, time delays do not matter and are omitted in the formulas. The sending rate and the basic link and source dynamics are given by

$$r_s = \frac{w_s}{\tau_s^0 + \tau_s} + \dot{w}_s$$

$$\dot{d}_\ell = \begin{cases} \frac{y_\ell}{c_\ell} - 1 & d_\ell > 0 \\ \max\left(0, \frac{y_\ell}{c_\ell} - 1\right) & d_\ell = 0 \end{cases}$$

Before describing the window update law, we need to consider how to model the congestion feedback. Each router marks packets independently, with some probability related to the current queue size. Since queueing delay d_ℓ is the state variable at link ℓ, we express the marking probability as $d_\ell/(d_\ell^0 + d_\ell)$, where the parameter d_ℓ^0 corresponds to q_0/c in the single link model. Assuming that packets are marked independently at each link, the probability that an ACK received by source s is marked is given by

$$1 - \prod_{\substack{\ell \\ A_{\ell s}=1}} \left(1 - \frac{d_\ell}{d_\ell^0 + d_\ell}\right)$$

To get this signalling to fit in the network utility maximization framework, we need linear aggregation via the routing matrix, in the form $p = A^T\lambda$. We get linear aggregation by defining the link price λ, not as the queue size, nor as the marking probability, but instead as:

$$\lambda_\ell = -\log\left(1 - \frac{d_\ell}{d_\ell^0 + d_\ell}\right) = \log\left(1 + \frac{d_\ell}{d_\ell^0}\right)$$

Then it follows that the probability that ACKs received by source s are marked is given by $1 - e^{-p_s}$. With this feedback signal, we can write down the window update law, as

$$\dot{w} = \frac{m_s - (1 - e^{-p_s})w_s}{\tau_s^0 + \tau_s}$$

The parameter m_s is the additive increase parameter, which we can think of as the packet size. The total system dynamics is then described by the following equations,

$$\dot{w} = \frac{m_s - (1 - e^{-p_s})w_s}{\tau_s^0 + \tau_s} \qquad \dot{d}_\ell = \begin{cases} \frac{y_\ell}{c_\ell} - 1 & d_\ell > 0 \\ \max\left(0, \frac{y_\ell}{c_\ell} - 1\right) & d_\ell = 0 \end{cases}$$

$$r_s = \frac{w_s}{\tau_s^0 + \tau_s} + \dot{w}_s \qquad \lambda_\ell = \log\left(1 + \frac{d_\ell}{d_\ell^0}\right)$$

together with the aggregation equations

$$y = Ar \qquad p = A^T\lambda \qquad \tau = A^T d$$

Next, we derive the equilibrium conditions. In equilibrium, the aggregation equations still holds, together with

$$m_s = \left(1 - e^{-p_s}\right) w_s \qquad y_\ell \le c_\ell$$

$$0 = d_\ell(c_\ell - y_\ell)$$

$$r_s = \frac{w_s}{\tau_s^0 + \tau_s} \qquad \lambda_\ell = \log\left(1 + \frac{d_\ell}{d_\ell^0}\right)$$

This does not quite fit the standard utility maximization problem:

$$\begin{aligned} \max_{r_s} \quad & \sum U_s(r_s) \\ \text{s. t.} \quad & r_s \geq 0 \\ & Ar \leq c \end{aligned}$$

since we have two different congestion signals, d_ℓ and λ_ℓ, that are propagated from links to sources.

It seems natural to treat the r_s as primal variables, and either d_ℓ or λ_ℓ as dual variables (since they are associated with the constraint $Ar \leq c$ and a complementary slackness condition $d_\ell(c_\ell - y_\ell) = 0$). But then the relation $\lambda_\ell = \log(1 + d_\ell/d_\ell^0)$ seems difficult to handle. Several other congestion control protocols proposed recently use both loss and delay as congestion signals [69, 112, 78]. The theory for this type of networked systems is not well-developed, although highly needed. An extension of the utility maximization framework to the case of several congestion signals would enable powerful techniques for analyzing such protocols over arbitrary topologies.

It is possible to use the standard framework if we make the following simplifications:

- Assume that d_ℓ^0 has the same value, d^0, at all links.
- Assume that $d_\ell \lll d^0$ for all ℓ.

Under these assumptions, we get

$$1 - e^{-p_s} \approx p_s \approx \frac{\tau_s}{d^0}$$

When p is eliminated in this way, τ can play the customary rôle as the dual variables associated with the capacity constraints.

We now derive a utility function from the equilibrium condition:

$$m_s = \left(1 - e^{-p_s}\right) w_s = \left(1 - e^{-p_s}\right) (\tau_s^0 + \tau_s) r_s \approx \frac{r_s}{d^0} \tau_s (\tau_s^0 + \tau_s)$$

Solving for τ_s gives

$$\tau_s = f_s(r_s) = \frac{\tau_s^0}{2} \left\{ \sqrt{1 + \frac{4 m_s d^0}{(\tau_s^0)^2 r_s}} - 1 \right\}$$

which clearly is a decreasing function in r_s. We find the utility function by integration, using

$$\int \sqrt{1 + \frac{2\alpha}{x}}\, \mathrm{d}x = x \sqrt{1 + \frac{2\alpha}{x}} + \alpha \log \frac{\sqrt{1 + \frac{2\alpha}{x}} + 1}{\sqrt{1 + \frac{2\alpha}{x}} - 1}$$

Put $\alpha = 2m_s d^0/(\tau_s^0)^2$ to get

$$\begin{aligned} U_s(r) &= \int_0^r f_s(r')\,\mathrm{d}r' \\ &= \frac{\tau_s^0}{2}\left\{ r\left(\sqrt{1+\frac{2\alpha}{r}}-1\right) + \alpha \log \frac{\sqrt{1+\frac{2\alpha}{r}}+1}{\sqrt{1+\frac{2\alpha}{r}}-1} \right\} \\ &= \frac{\tau_s^0}{2}\left\{ r\sqrt{1+\frac{4m_s d^0}{(\tau_s^0)^2 r}} - r + \frac{2m_s d^0}{(\tau_s^0)^2} \log \frac{\sqrt{1+\frac{4m_s d^0}{(\tau_s^0)^2 r}}+1}{\sqrt{1+\frac{4m_s d^0}{(\tau_s^0)^2 r}}-1} \right\} \end{aligned}$$

To understand the shape of this function, we can look at the asymptotics when r is very small or very large. We get

$$\begin{aligned} U_s(r) &= \tau_s^0\sqrt{2\alpha r} + O(r) = \sqrt{2m_s d^0 r} + O(r) && \text{for small } r \\ U_s(r) &= \frac{\tau_s^0}{2}\alpha \log r + O(1) = \frac{m_s d^0}{\tau_s^0}\log r + O(1) && \text{for large } r \end{aligned}$$

We see that for small r, $U_s(r)$ grows rapidly, as $\sqrt{r}$. This seem related to the condition that $w_s > m_s$. For large r, the utility grows as the $\log r$. This utility is the same as in TCP Vegas and Fast TCP, and corresponds to proportional fairness.

Since we had to use some approximations when deriving this utility function from the equilibrium, we can not expect it to describe the fairness properties exactly. However, we can still conclude that the sending rates in equilibrium should be close to proportionally fair.

The weight m_s/τ_s^0 implies that flows with larger packet size or shorter propagation will get higher throughput, just as for TCP New Reno. This dependence on packet size and propagation delay can be seen already from the equilibrium equation: Since $w_s = m_s/p_s$ in equilibrium, if some flows share the same set of bottlenecks, so that they see the same price p_s, then they all have the same window size measured in units of packets. And the equilibrium sending rates are $w_s/(\tau_s^0+\tau_s) = m_s/(p_s\,(\tau_s^0+\tau_s))$, i.e., proportional to m_s and inversely proportional to the equilibrium RTT.

6.7 Packet-level simulations

The fluid flow model neglects important packet-level features of the proposed mechanism, in particular, it does not capture the limited information about the mark probability p. To see how the mechanism behaves at the packet level, the mechanism has been implemented in the `ns2` simulator. The implementation consists of a couple of small changes:

- Two new bits in the `ns2` packet header, analogous to the ECN and ECN echo bits.

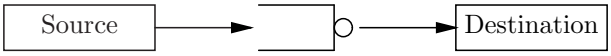

Figure 6.11: The simplest possible topology. A single data flow, with a single intermediate bottleneck router.

- The `Agent/TCPSink` class handles reception of TCP data and sending of ACK packets. It is extended to copy the new mark bit from each TCP data packets to the corresponding ACK.
- The `Queue/DropTail` class implements the simplest queueing discipline, drop tail. This class is extended with a new attribute q_0. The new marking scheme is enabled, by setting q_0 to a positive value. When enabled, packets are marked at arrival with a probability $q/(q_0 + q)$.
- The `Agent/TCP/Newreno` class implements the sending side of TCP New Reno. It is modified to examine the echoed mark bit of received ACKs, and to decrease current window size by one packet for each received mark.

Neither the slow start phase or New Reno's reaction to packet losses is modified at all. To force the TCP sender to leave the slow start phase, the initial slow start threshold in all the simulations were set to a value close to the product of the RTT and the fair share.

Single bottleneck, single flow

The first scenario uses a single TCP flow over a single bottleneck, this topology is shown in Fig. 6.11. The bottleneck capacity is 2 Mbit/s, and the roundtrip propagation delay (excluding queueing delay) is 100 ms. This implies a bandwidth delay product of 16 packets, and the buffer size is set to 21 packets. The simulation runs for 60 s, with a file transfer starting at time 0.1 s. The bottleneck link is shared with Poisson cross-traffic, 20% of the capacity, increased to 40% during the interval 20–40 s.

This scenario is simulated twice, once with New Reno, once with the new protocol, with $q_0 = 32$, twice the bandwidth-delay product. The results for a segment of time around the cross-traffic change (at 20 s) is shown in Figs 6.12 and 6.13.

Table 6.1 shows average properties. The new mechanism almost eliminates packet losses, while at the same time the TCP throughput is increased and the queueing delay is reduced.

Single bottleneck, two flows

For the next scenario, we keep the same 2 Mbit/s bottleneck link with 20% Poisson cross-traffic, and start two flows in parallel. The topology is illustrated in Fig. 6.14. The first flow has a roundtrip propagation time (excluding queueing delay) of 20 ms,

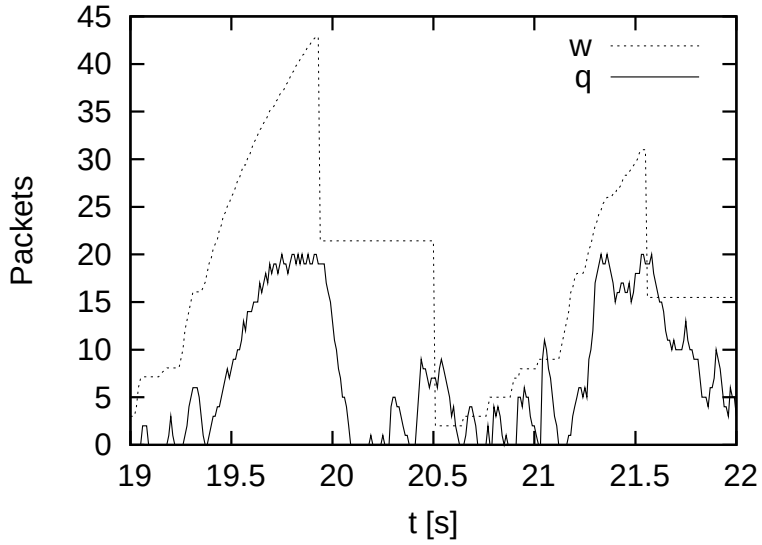

Figure 6.12: Single bottleneck, single flow. We see the congestion window and the queue size for a New Reno flow. At 20 s, the cross-traffic is increased from 20% to 40%.

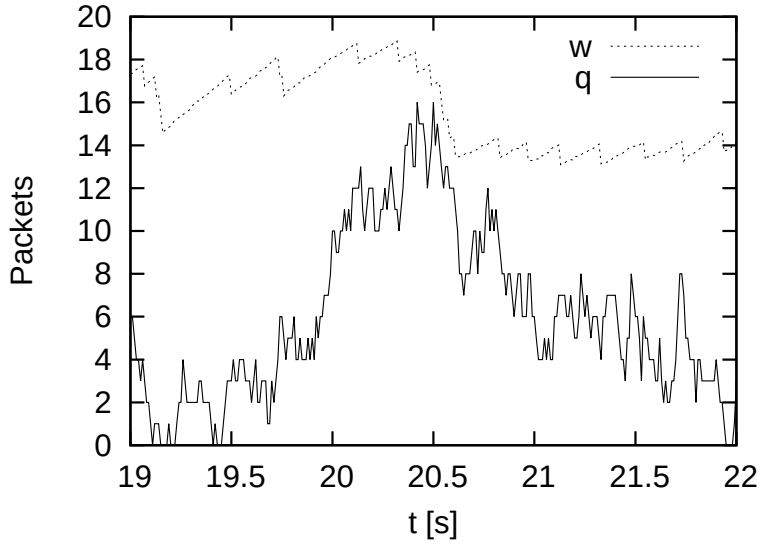

Figure 6.13: Single bottleneck, single flow. We see the congestion window and the queue size for a flow using the new protocol. At 20 s, the cross-traffic is increased from 20% to 40%.

	New Reno	New protocol
Loss rate (%)	4.27	0.02
Utilization (Mbit/s)	1.64	1.90
TCP throughput (Mbit/s)	1.02	1.40
Queue average (packets)	7.26	3.77
std. dev. (packets)	7.12	3.41
Forward delay average (ms)	114.66	80.84
std. dev. (ms)	40.11	19.61

Table 6.1: Average quantities for the single flow case.

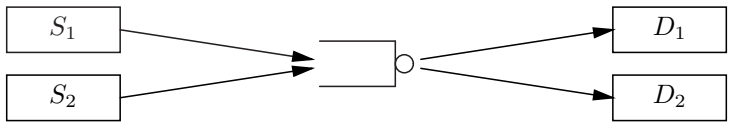

Figure 6.14: Topology with a single bottleneck shared by two flows.

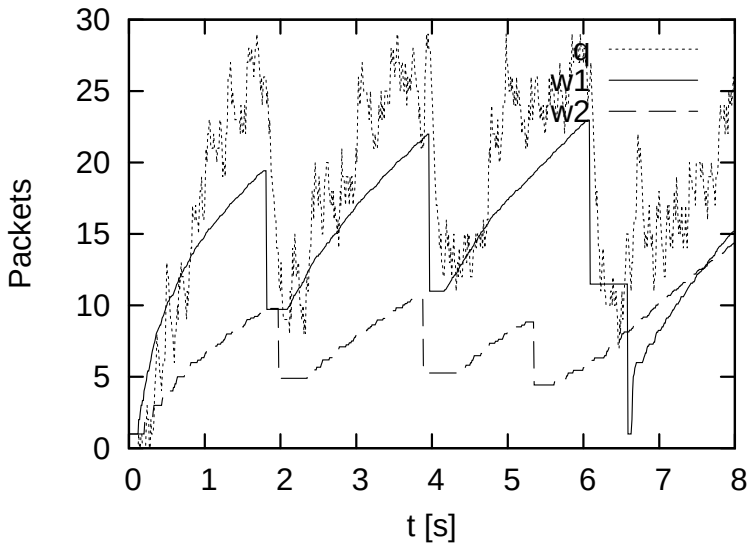

Figure 6.15: Single bottleneck with two New Reno flows.

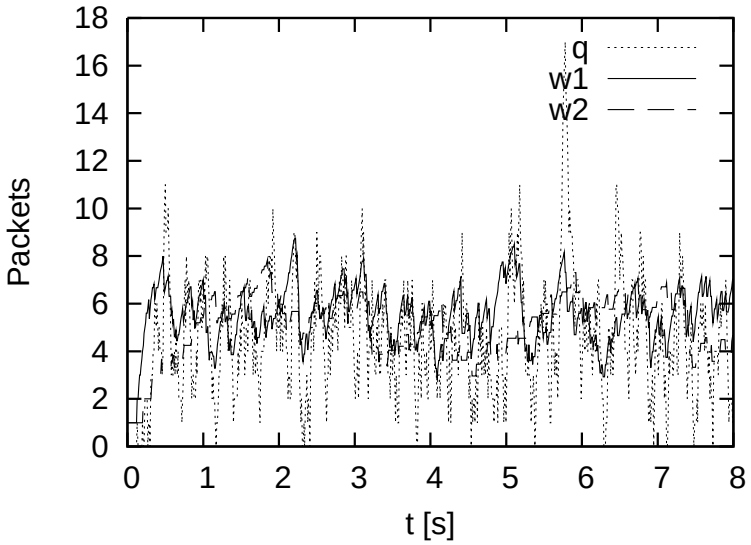

Figure 6.16: Single bottleneck with two flows using the new protocol.

	New Reno	New protocol
Loss rate (%)	1.56	0.00
Utilization (Mbit/s)	2.00	1.99
Queue average (packets)	19.27	4.58
std. dev. (packets)	6.00	2.32

Table 6.2: Average properties for the link in the two-flow scenario.

	New Reno		New protocol	
RTT (ms)	20	100	20	100
TCP throughput (Mbit/s)	1.22	0.38	1.17	0.42
TCP window (packets)	14.96	7.77	5.81	5.21
std. dev.	4.37	2.95	1.16	1.20
Forward delay average (ms)	135.55	178.34	45.58	88.53
std. dev. (ms)	37.33	35.61	13.76	13.41

Table 6.3: Average properties per flow, for the two-flow scenario.

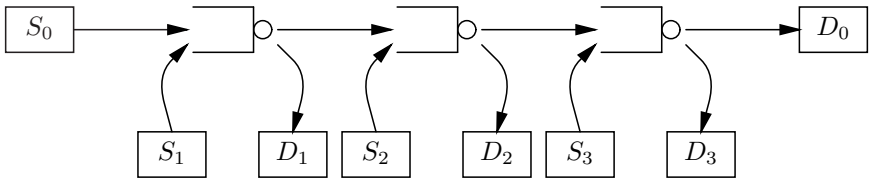

Figure 6.17: The "parking-lot" topology, with three bottlenecks in series.

and the second flow has a five times larger delay, 100 ms. The buffer size is set to 30 packets, and q_0 is set to 15 packets. The evolution of the queue size and the two window sizes are shown in Fig. 6.15–6.16.

Average quantities are summarized in Tab 6.2 and Tab. 6.3. The sharing of capacity is roughly the same for the new protocol and New Reno, the flow with the longer delay gets approximately a third of the throughput of the flow with the shorter delay. For both the new protocol and TCP New Reno, flows that share the same set of bottlenecks are expected to have the same equilibrium window size. In this experiment, that is not the case for the two New Reno flows, and there also seems to be more variation in the window size when using New Reno.

"Parking-lot" topology

In the final scenario, we have three bottleneck links in series, illustrated in Fig. 6.17. The links are still 2 Mbit/s, with both the buffer size and q_0 is set to 20 packets. Over each link, there is one TCP flow, and 20% Poisson cross-traffic. These short flows have 20 ms RTT excluding queueing. We also add one long RTT flow, traversing all three bottlenecks, with a RTT of 150 ms excluding queueing.

The evolution of queues are shown in Fig. 6.18–6.19, while the window sizes are shown in 6.20–6.21. The average properties are summarized in Tab. 6.4–6.5.

6.8 Summary

We have proposed a new congestion control protocol, consisting of the standard ACK-clock based inner-loop, and a novel outer-loop that adjusts the window size. Routers mark packets with a probability depending on their instantaneous queue size. In the absence of marks on received ACKs, sources increase their window sizes linearly. When a marked ACK is received, the source subtract the size of the corresponding data packet from its congestion window.

The rule of reducing window size by one packet on the reception of a marked ACK is a subtle change from standard ECN processing, which has two important benefits: The frequency of packet marks can be much higher than with standard AQM/ECN schemes, on average, a flow in equilibrium will get receive one marked ACK per RTT. In this way, sources are provided with more information about the network state. Furthermore, from a router's point of view, the effect of the decision

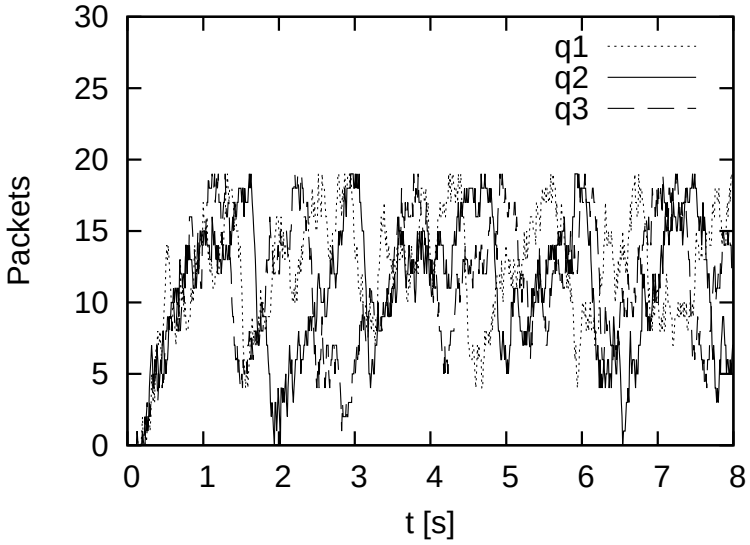

Figure 6.18: Evolution of the three bottleneck queues in the parking lot scenario, with New reno flows.

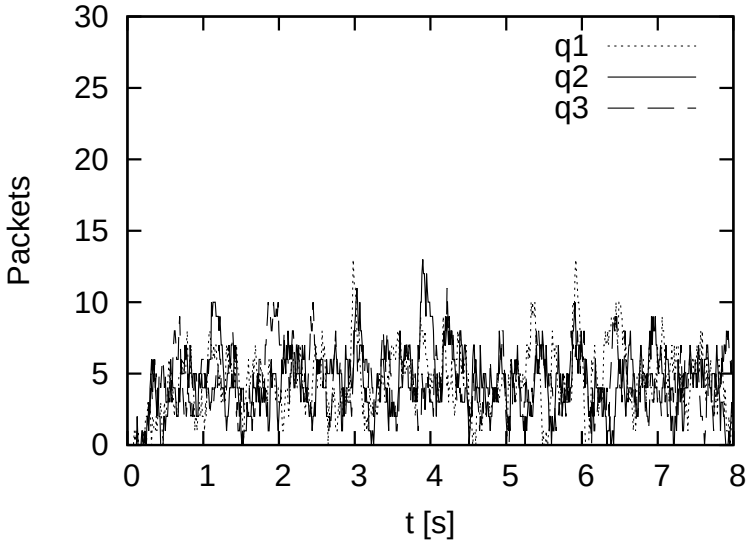

Figure 6.19: Evolution of the three bottleneck queues in the parking lot scenario, with the new protocol.

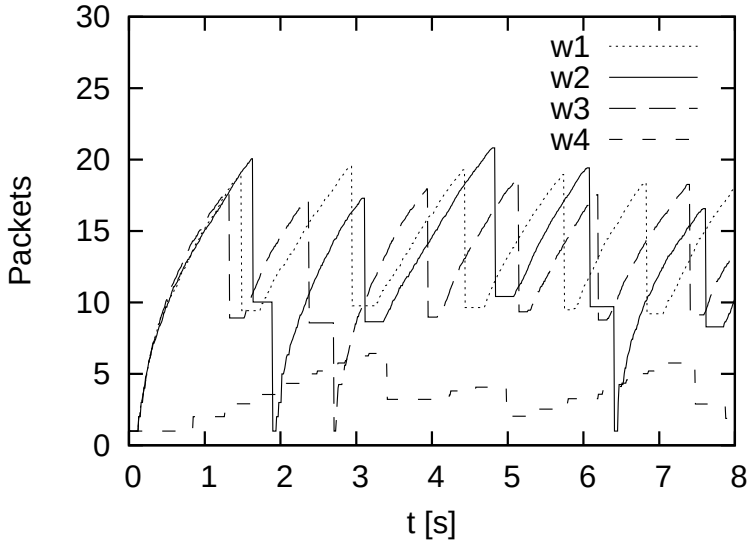

Figure 6.20: Window sizes for the four flows in the parking lot scenario, with New Reno flows.

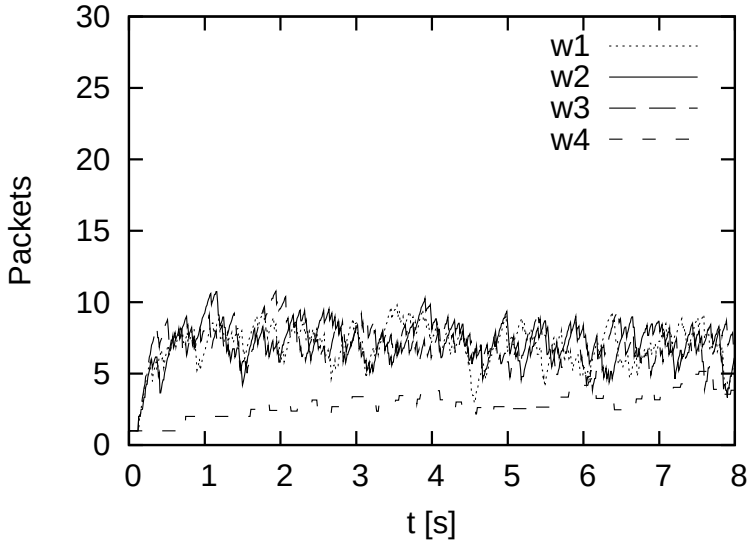

Figure 6.21: Window sizes for the four flows in the parking lot scenario, with the new protocol.

	New Reno			New protocol		
Loss rate (%)	1.96	1.99	1.67	0.00	0.00	0.00
Utilization (Mbit/s)	2.00	2.00	2.00	1.99	1.99	1.99
Queue average (packets)	11.98	12.07	11.94	4.37	4.43	4.33
std. dev. (packets)	4.32	4.37	4.35	2.15	2.16	2.09

Table 6.4: Average properties for the three bottlenecks in the parking-lot scenario.

	New Reno		New protocol	
Flow	#0	#1	#0	#1
TCP throughput (Mbit/s)	0.11	1.49	0.12	1.48
TCP window (packets)	4.49	13.05	2.94	7.06
std. dev.	2.01	3.57	0.75	1.26
Forward delay average (ms)	323.05	91.29	181.30	44.63
std. dev. (ms)	46.30	26.29	23.59	12.49

Table 6.5: Average properties for the flows, in the parking-lot scenario. Flow #0 is the long flow, flow #1 is one of the short flows. The values for flow #2 and #3, and are omitted in this table.

to mark a packet becomes much more predictable, since it does not depend on the unknown window size of the corresponding flow.

The proposed marking scheme uses a single parameter, q_0, per link. The value should be on the same order as the expected value of the bandwidth-delay product of flows using the link. We believe tuning of this parameter should be fairly easy. Larger values results in a larger equilibrium queue size and a more stable system, while smaller values result in a smaller equilibrium queue size. For small values of q_0, the system is unstable, but in our simulation, this instability corresponds to small oscillations of the queue size and queue underflow, which indicates that the penalty for choosing a too small q_0 is reduced link utilization, not large queue oscillations.

The proposed protocol has been implemented in `ns2`, and simulated over a single link topology, and over a "parking-lot" topology with several bottlenecks. The simulations show that the proposed protocol maintains full utilization and small queueing delays. Furthermore, each flow gets a share of capacity close to what the flows would get if they were all using TCP New Reno.

Further work

The development of the new congestion control protocol in this chapter is not complete. Some interesting problems are:

Aggregation Investigate the equilibrium and dynamical properties when a large number of flows share a bottleneck.

Slow start In TCP New Reno, the first transition from the slow start state to congestion avoidance state happens after the first packet loss. If packet losses are eliminated, this transition must be based on some other signal. We need a slow-start like mechanism that can grow the window size quickly when the network is uncongested, and, based on received packet marks, switch to congestion avoidance before the bottleneck queue grows large.

Traffic shaping The sender could use some basic traffic shaping, e.g., keep a minimum inter-packet interval corresponding to half the estimated RTT, divided by the current window size in packets. The effect on both the inner-loop, in particular for flows with heterogeneous RTTs, and the outer-loop, needs further investigation.

Wireless links Investigate the behavior over wireless links, characterized by time varying capacity, random delays, and non-congestion packet losses.

Chapter 7

Cross-layer radio network feedback

WIRED links are characterized by a small loss rate, short constant propagation delay, and constant (and often high) capacity. As described in Sec. 2.7, wireless links differ not only in the average of these characteristics, but also in that all of the delay, loss rate, and capacity, are time varying. To some degree, the link-layer design enables tradeoffs between these characteristics, e.g., introducing link-layer retransmissions reduces the loss rate, at the cost of larger and more varying delays.

In this chapter, we focus on handling the capacity variation when using TCP/IP over the cellular telephony system. Standard TCP adapts quite slowly to capacity variation, and in order to get high utilization of available capacity, it is necessary to use a large buffer in front of the radio link. Large RTTs, which are common in these systems, also hurts performance, in particular for short flows.

To improve performance, we consider cross layer signalling from the link layer at the radio link, to the transport layer of the TCP sender. In this chapter, we consider file transfers in the down-link, i.e., from a server in the Internet to the mobile terminal. Sec. 7.1 gives an overview of the High-Speed Downlink Packet Access (HSDPA). The system architecture, with a proxy and cross-layer signalling, is described in Sec. 7.2. The control structure, with feedforward of the channel capacity and feedback of the RNC queue size, is described in Sec. 7.3. In Sec.7.4, we prove that the proposed control mechanisms give a stable system. In Sec. 7.5, we present simulations studies, for both a dedicated channel, and a shared HSDPA channel. We also compare the performance gain to what one would get with other widely known proposals for improved TCP over wireless, Snoop and Eifel. Finally, in Sec. 7.6, we discuss the results and the implications for the tradeoff between throughput and delay.

7.1 High-speed downlink packet access

The High-Speed Downlink Packet Access (HSDPA) is an evolution of the third generation cellular network specified by the 3GPP [1, 2]. HSDPA allows theoretical downlink peak data rates of 14.4 Mbit/s, compared with 2 Mbit/s of UMTS (Universal Mobile Telecommunications Service).

In HSDPA three new channel types have been defined: the High-Speed Downlink Shared Channel (HS-DSCH), the High-Speed Shared Control Channel (HS-SCCH), and the High-Speed Dedicated Physical Control Channel (HS-DPCCH). HS-DSCH is the transport channel used to carry users' data. Its resources can be shared among all active HSDPA users in the cell. HS-DSCH is mapped onto a pool of physical channels denominated High-Speed Physical Downlink Shared Channels (HS-PDSCHs), which are multiplexed both in time and in code. Time is handled in Transmission Time Intervals (TTIs) of 2 ms and the code uses a constant spreading factor of 16, with a maximum of 15 for HS-PDSCHs. These channels may be all assigned to one terminal during the TTI, or may be split between several users. HS-SCCH is a downlink signalling channel used to carry information between the base station and the terminal before the beginning of each scheduled TTI. HS-DPCCH is an uplink low bandwidth signalling channel used to carry positive or negative acknowledgments and channel quality indicator, which provides information related to the wireless link bandwidth.

The main features introduced by HSDPA concern the use of an Adaptive Modulation and Coding (AMC) scheme, of a fast Hybrid-Automatic Repeat reQuest (HARQ) mechanism, and of a fast scheduling.

The scheduler determines the terminal (or terminals) to which the HS-DSCH should transmit and, together with AMC, the data rate to be used. The scheduling algorithms are Round Robin, which schedules users with a first-in first-out approach, Maximum Carrier to Interference (Max C/I), which schedules only users that are experiencing the Max C/I during that TTI, and Proportional Fair, which provides a trade-off mechanism between Round Robin and Max C/I. The Round Robin algorithm involves high fairness among all users, but it may cause a reduction of the overall system throughput since users may be served even when they are experiencing bad channel conditions. On the other hand, Max C/I provides the maximum overall throughput for the system but it causes unfairness among users, penalizing those located at the cell edge.

When used for IP transport, the HSDPA system appears as a link characterized by time-varying capacity, and random delays.

7.2 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: A Radio Network Controller (RNC), which among other

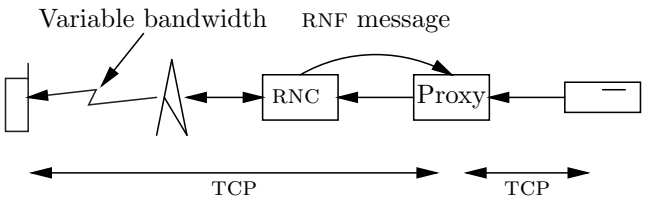

Figure 7.1: Radio network feedback architecture. The mobile terminal on the left downloads a file from the server on the right, via the proxy. During the transfer, the RNC generates cross-layer RNF messages including information about the current capacity over the radio link, and the current RNC queue length. The proxy uses this information to adjust its window size.

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 7.1, 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 Radio Network Feedback (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 TCP connection between the terminal and the server.

We use the following notation. The time-varying bandwidth of the radio link is c. The roundtrip propagation delay, excluding queueing at the RNC, is denoted τ, and the "pipe size", or bandwidth-delay product, is $c\tau$. The queue length at the RNC is denoted q. The current window size at the proxy is denoted w.

7.3 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 information about the current capacity and current queue size. The control structure, with feed-

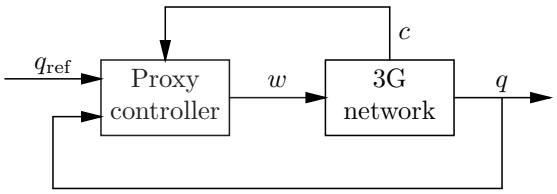

Figure 7.2: Control structure. The controller uses cross-layer signalling: Feedforward of the available radio link bandwidth c, and feedback of the RNC queue length q. Note that both the RNC and the proxy belong to the operator's administrative domain.

forward of radio bandwidth c, and feedback of queue length q, is shown in Figure 7.2.

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. For a dedicated channel, the capacity switches between discrete values, i.e., the capacity signal is piecewise constant. The generation of the bandwidth update messages is event-triggered. For a shared HSDPA channel, the allocation of capacity is more complicated and is depends on several random factors, such as the behavior and movement of other users. Event-triggered generation of bandwidth update messages could be based on a change-detection algorithm running at the RNC. For the current work, however, the bandwidth update messages for the HSDPA channel is based on regular sampling, where the transmitted value is simply the average capacity during the previous sampling period.

When the controller receives a bandwidth update message, it resets the window size to a new value,

$$w = \widehat{c}\widehat{\tau} + q_{\text{ref}} \tag{7.1}$$

Here $\widehat{\tau}$ is an estimate of the propagation delay τ, and $\widehat{c}$ is the new capacity from the RNC's message.

When the interval between bandwidth updates is long, i.e., in the dedicated channel scenario, 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,

$$w_{j+1} = w_j + \beta\,(q_{\text{ref}} - q_{j+1}) \tag{7.2}$$

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.

7.4 Stability analysis

Feedforward control

For the dynamics of the queue, subject to the ACK-clock inner-loop, we use the joint link model from Chapter 3,

$$\dot{q}(t) = \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c$$

As usual, we also have a non-negativity constraint. The above equation is valid only when either $q(t) > 0$ or the right hand side is non-negative; otherwise, $\dot{q}(t) = 0$. We consider the case of γ close to 1, i.e., a small amount of cross-traffic.

If we substitute the feed-forward controller (7.1), this gives

$$\dot{q}(t) = \frac{\widehat{c}\widehat{\tau} + q_{\text{ref}}}{\tau + q(t-\tau)/c} - \gamma c$$

This dynamical system is the same as the one studied in Chapter 4. By Theorem 4.2, the system is globally asymptotically stable, in the sense that for any initial state, if c is kept constant, then $q(t) \to q^*$ as $t \to \infty$, where q^* is the equilibrium:

$$q^* = \max\left(0, \frac{q_{\text{ref}}}{\gamma} + \frac{\widehat{c}\widehat{\tau}}{\gamma} - c\tau\right)$$

Furthermore, by Prop. 4.5, the convergence time is bounded by $3.4(\tau + q^*/c)$. We also see the effect of uncertainty in γ and in the estimates $\widehat{c}$ and $\widehat{\tau}$: we have a bias

$$q^* - q_{\text{ref}} = \left(\frac{1}{\gamma} - 1\right) q_{\text{ref}} + \frac{\widehat{c}\widehat{\tau}}{\gamma} - c\tau$$

Feedback control

We now close the loop, using queue length samples. The primary objective of the feedback control is to cancel the bias that results when using feedforward control from estimates $\widehat{c}$ and $\widehat{\tau}$ that are uncertain, or when $\gamma < 1$. A natural approach is to use an integrating controller, and the feedback control law

$$\dot{w}(t) = -k\left(q(t-\tau) - q_{\text{ref}}\right) \tag{7.3}$$

where the gain k is a control parameter. For simplicity, we use the roundtrip propagation delay τ here, even though the one-way signalling delay from RNC to the proxy may be significantly shorter than τ.

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

$$\begin{cases} \dot{q}(t) = \dfrac{w(t-\tau)}{\tau + q(t-\tau)/c} - k\left(q(t-\tau) - q_{\text{ref}}\right) - \gamma c \\ \dot{w}(t) = -k\left(q(t-\tau) - q_{\text{ref}}\right) \end{cases} \tag{7.4}$$

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

$$\dot{w}(t) = \max\Big(0, -k\left(q(t-\tau) - q_{\text{ref}}\right)\Big)$$

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

Equations (7.4) have an equilibrium point given by $q^* = q_{\text{ref}}$ and $w^* = \gamma(c\tau + q_{\text{ref}})$. The RTT in stationarity is $\tau^* = \tau + q_{\text{ref}}/c$. Let $\widetilde{q}(t) = q(t) - q^*$ and $\widetilde{w}(t) = w(t) - w^*$. With this notation, the system can be written as

$$\begin{cases} \dot{\widetilde{q}}(t) = \dfrac{\widetilde{w}(t-\tau) - \gamma\widetilde{q}(t-\tau)}{\tau^* + \widetilde{q}(t-\tau)/c} - k\widetilde{q}(t-\tau) \\ \dot{\widetilde{w}}(t) = -k\widetilde{q}(t-\tau) \end{cases}$$

Let us first address local stability. Linearization of the system around the equilibrium gives

$$\begin{cases} \dot{\widetilde{q}}(t) = -\left(k + \dfrac{\gamma}{\tau^*}\right)\widetilde{q}(t-\tau) + \dfrac{\widetilde{w}(t-\tau)}{\tau^*} \\ \dot{\widetilde{w}}(t) = -k\widetilde{q}(t-\tau) \end{cases}$$

Then Theorem 4.18 implies that the system is stable provided that

$$\begin{aligned} \tau k &\leq \frac{\pi}{2}\,\frac{q_{\text{ref}} + (1 - 2\gamma/\pi)c\tau}{q_{\text{ref}} + (1 + 4/\pi)c\tau} \\ \tau k(c\tau - q_{\text{ref}}) &\leq \gamma c\tau \end{aligned}$$

As one might have expected, the stability condition is in the form of a bound on τk, the product of the system delay and gain. Furthermore, the bound depends on γ and the fraction $q_{\text{ref}}/(c\tau)$. And finally, we can conclude that the system is stable for all sufficiently small k, in particular, $\tau k \leq \min(\gamma, 0.25)$ is sufficient.

Next, we address global stability. We need to simplify the model slightly. We have seen in Sec. 3.5 that the alternative model for the queue dynamics, without signalling delays, is a reasonable approximation. So we omit those delays. We also omit the feedback delay in the outer-loop; this is in general not a good approximation, but it can be justified in the case that the signalling delay from RNC to proxy is small in comparison to the system dynamics. With these simplifications, we get the non-linear system

$$\begin{cases} \dot{\widetilde{q}}(t) = \dfrac{\widetilde{w}(t) - \gamma\widetilde{q}(t)}{\tau^* + \widetilde{q}(t)/c} - k\widetilde{q}(t) \\ \dot{\widetilde{w}}(t) = -k\widetilde{q}(t) \end{cases}$$

Consider the Lyapunov function

$$V(\widetilde{q}, \widetilde{w}) = \left(\frac{\tau^*}{2} + \frac{\widetilde{q}}{3c}\right)\widetilde{q}^2 + \frac{1}{2k}\widetilde{w}^2$$

Note that

$$\frac{\tau^*}{2} + \frac{\widetilde{q}}{3c} = \frac{\tau + q_{\text{ref}}/c}{2} + \frac{q - q_{\text{ref}}}{3c} \geq \frac{\tau}{2} + \frac{q_{\text{ref}}}{6c} > 0$$

so that V is positive definite on the domain $\widetilde{q} \geq -q^* = -q_{\text{ref}}$ and $\widetilde{w} \geq -w^* = -(c\tau + q_{\text{ref}})$. We get

$$\begin{aligned}
\dot{V}(\widetilde{q}, \widetilde{w}) &= \widetilde{q}^2 \left(\tau^* + \frac{\widetilde{q}}{c}\right) \dot{\widetilde{q}} + \frac{1}{k} \widetilde{w} \dot{\widetilde{w}} \\
&= -\widetilde{q}^2 \left(\gamma + k(\tau^* + \widetilde{q}/c)\right) \\
&\leq -\widetilde{q}^2 \left(\gamma + k\tau\right)
\end{aligned}$$

We also need to consider the case of active non-negativity constraints on the borders. The constraint on w is active when $\widetilde{w} = -w^*$ and $\widetilde{q} > 0$. In this case $\dot{w} = 0$ and

$$\dot{\widetilde{q}} = \frac{-w^* - \gamma \widetilde{q}}{\tau^* + \widetilde{q}/c} - k\widetilde{q} < 0$$

and it follows that $\dot{V} < 0$. The constraint on q is active when $\widetilde{q} = -q_{\text{ref}}$ and

$$\frac{\widetilde{w} + \gamma q_{\text{ref}}}{\tau} + k q_{\text{ref}} < 0$$

This implies that $\widetilde{w} < 0$. Furthermore $\dot{\widetilde{q}} = 0$ and $\dot{\widetilde{w}} = k q_{\text{ref}} > 0$. It follows that $\dot{V} < 0$ also in this case.

By LaSalle's theorem, we can conclude that for any initial condition $\widetilde{w}(0) \geq -w^*$ and $\widetilde{w}(0) \geq -q^*$, the trajectory converges to some invariant set of the line $\widetilde{q} = 0$, and the only such set is the origin. .

Sampled feedback

In a practical implementation, the continuous signal $q(t)$ will not be available at the proxy; the RNC has to sample the signal and send the sample values as RNF messages to the proxy. Recall that the objective of the feedback loop is to eliminate the bias caused by uncertainty in the estimates $\widehat{c}$ and $\widehat{\tau}$. If we select a sampling time that is large in comparison to the RTT, the following simple feedback law, corresponding to the choice $\beta = 1$ in Eq. (7.2), eliminates the bias rapidly, and within two sampling periods in the case $\gamma = 1$.

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

If the sampling time is large compared to the RTT, then the sampling time is also a couple of times longer than the convergence time constant. We get

$$q_{j+1} \approx \frac{w_j}{\gamma} - c\tau \qquad \text{Convergence}$$

$$w_{j+1} = w_j + q_{\text{ref}} - q_{j+1} \approx c\tau + q_{\text{ref}} - \left(\frac{1}{\gamma} - 1\right) w_j \qquad \text{Feedback}$$

If we put $w^* = \gamma(c\tau + q_{\text{ref}})$, this can be rearranged as

$$q_{j+1} - q_{\text{ref}} \approx \frac{w_j - w^*}{\gamma}$$
$$w_{j+1} - w^* = -\left(\frac{1}{\gamma} - 1\right)(w_j - w^*)$$

or

$$w_j - w^* \approx \left(1 - \frac{1}{\gamma}\right)^j (w_0 - w^*)$$
$$q_{j+1} - q_{\text{ref}} \approx \frac{1}{\gamma}\left(1 - \frac{1}{\gamma}\right)^j (w_0 - w^*)$$

In particular, if $w_0 = \widehat{c}\widehat{\tau} + q_{\text{ref}}$, as set by the feedforward controller, then

$$\begin{aligned} q_2 &\approx q_{\text{ref}} - \frac{1}{\gamma}\left(\frac{1}{\gamma} - 1\right)(\widehat{c}\widehat{\tau} + q_{\text{ref}} - \gamma(c\tau + q_{\text{ref}})) \\ &= q_{\text{ref}} - \frac{1}{\gamma}\left(\frac{1}{\gamma} - 1\right)(\widehat{c}\widehat{\tau} - \gamma c\tau) - \left(\frac{1}{\gamma} - 1\right)^2 q_{\text{ref}} \end{aligned}$$

We see that q_j and w_j converge as powers of the small number $(1/\gamma - 1)$. When $\gamma = 1$, this means that q converges to q_{ref} in two samples.

7.5 Simulation studies

To quantify how the RNF proxy improves performance, compared to end-to-end TCP Reno, we have performed a series of simulations studies.

- Our first study considers a dedicated channel model, where the channel is subject to stepwise changes to the capacity, and to temporary outages. Such channel variations are typical for channels in UMTS.

- The second simulation study considers the shared download channel in HSDPA. The capacity given to each user is decided by the Radio Network Controller, based on the varying system load and the radio conditions for individual users. For this study, the capacity variation is based on data, provided by Ericsson, from detailed radio-level simulations.

- The third simulation study also focuses on HSDPA. The objective here was not just to compare performance to TCP Reno, but to compare the performance gain to the gain from other proposed mechanisms in the literature, in particular Snoop and Eifel.

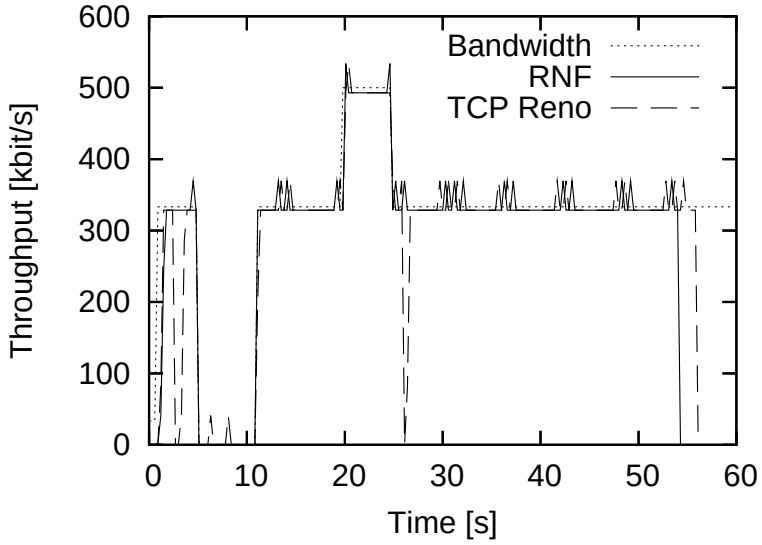

Figure 7.3: Resulting throughput for the dedicated channel scenario. The dotted line is the time-varying bandwidth to the terminal, the dashed line is the throughput for TCP Reno, and the solid line is the throughput using the proposed RNF controller.

Dedicated channel

The dedicated channel is characterized by a bandwidth that varies between discrete values, depending on the current radio conditions. In this simulation, we use a link with a nominal bandwidth of 1 Mbit/s, which is subject to an outage of 6 s, starting at time 5 s, and an excellent radio environment where the bandwidth is increased to 1.5 Mbit/s, for 5 s starting at time 20 s. The roundtrip propagation delay between proxy and terminal is 223 ms. The packet size is 1500 bytes, and the RNC buffer size is 30 packets, corresponding to the bandwidth-delay product during the time interval with the largest bandwidth.

Figure 7.3 shows the resulting throughput as a function of time, and Fig. 7.4 shows the evolution of the RNC queue. In Fig. 7.3, note that the file download is completed 2 s earlier, which is a modest improvement of user response time. In Fig. 7.4, we see that the queue stays small, with small fluctuations. Due to errors in the RTT estimate, the feedforward from the bandwidth changes results in a queue size slightly different from the target value $q_{\text{ref}} = 6$, but this error is corrected by the feedback from the actual queue size.

The `ns2` implementation of the RNF proxy, which is used in this and the other simulation scenarios, was developed as part of the Master of Science thesis [90].

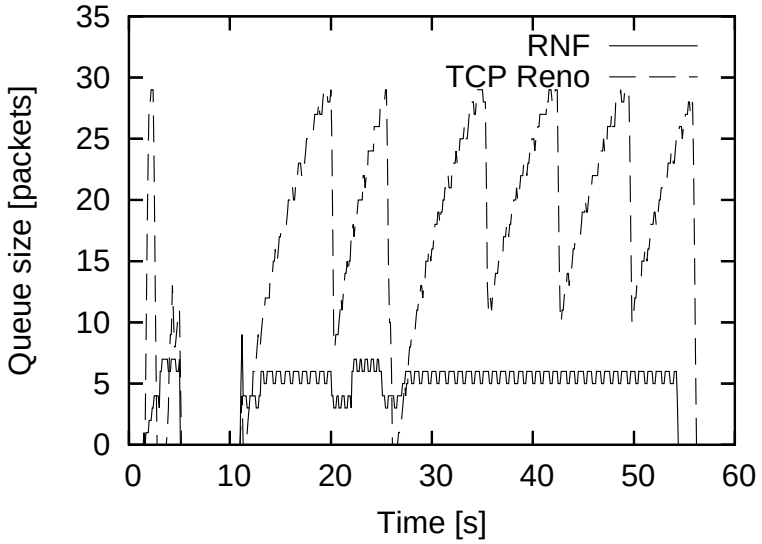

Figure 7.4: Resulting RNC queue size for the dedicated channel scenario. The dashed line is the queue size with TCP Reno, and the solid line is the queue size with the RNF controller. With RNF, the queue stays small, with small fluctuations.

HSDPA shared channel scenario

To simulate a realistic scenario of how the rate fluctuates for an end-user it is important that all the aspects which are impacting the end-user rate are considered. As described in Sec. 7.1 the rate fluctuates due to variations in radio quality and cell load. The load variation in number of users depends on statistical variations in call originations but also due to mobility when users move between cells.

We use a two stage simulation: First, the radio system is simulated in a fairly loaded scenario, where users move around and download files using TCP/IP. From this radio level simulation, we extract the available bandwidth per user in one of the cells, which is a time varying signal. The second stage of the simulation implements a link with this particular bandwidth variation in the `ns2` network simulator [94], for evaluation of the RNF mechanism.

The setup for the radio layer simulation was as follows: 7 sites and 21 cells, 1500 m site-to-site distance, realistic radio and fading models, and end-users moving at 1 m/s while downloading files of 50 Kbyte via TCP/IP. The output result from the radio layer simulations, the end-user rate level and fluctuation, is of course different depending on how high system load that is used in the simulation. The result also depends on the services that are used. However, we believe that on the time scales that matter for TCP-traffic, the character of the bandwidth variations are similar for a wide range of load levels and services.

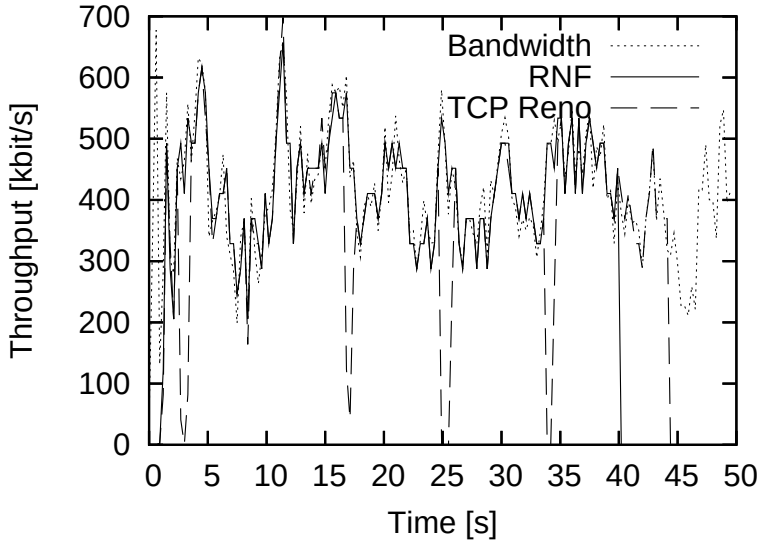

Figure 7.5: Resulting throughput for the HSDPA scenario. The dotted line is the time-varying bandwidth to the terminal, the dashed line is the throughput for TCP Reno, and the solid line is the throughput using the proposed RNF controller.

Since the bandwidth varies continuously, there are no distinct stepwise changes of the bandwidth. In this scenario, we base the bandwidth feedforward on regular sampling instead of the event-based signalling in the previous scenario. We use a sample time of 0.5 s, and for each sample, the RNC calculates the averaged bandwidth available to the user during the previous 0.5 s interval. The queue feedback is omitted—it was intended to operate on a time scale faster than the feedforward control and slower than the RTT, and is not applicable here.

Figure 7.5 shows the resulting throughput as a function of time, and Fig. 7.6 shows the evolution of the RNC queue. The results are similar to the case of a dedicated channel: The RNC queue stays small, and the total download time is a few seconds shorter than with TCP Reno.

Comparison to Eifel and Snoop

Finally, we compare the performance gain with the RNF proxy, to the performance gain with Eifel and Snoop.

In these simulations, the radio channel is modelled using the EURANE extension to `ns2`, developed by Ericsson Netherlands and other partners within the European 5th framework project SEACORN [98, 94]. The simulation scenario considers the connection to a user located 450 m from a base station. There are 29 other users

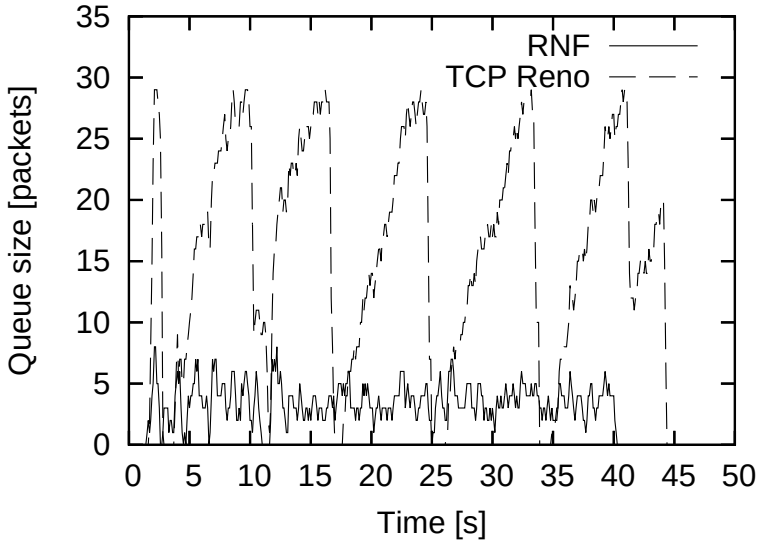

Figure 7.6: Resulting RNC queue size for the HSDPA scenario. The dashed line is the queue size with TCP Reno, and the solid line is the queue size with the RNF controller. Like in the dedicated channel scenario, the RNF controller results in a small queue.

competing for the channel, with distance to the base station varying from 50 m to 750 m. User mobility follows the standard "Pedestrian A" scenario, with pedestrian users, walking with a velocity of 3 Km/h. The RNC uses the Max C/I scheduler, and the sample period for the RNF capacity feedback is 250 ms. The path from web server to the base station has a bottleneck of 10 Mbit/s outside of the operator's network, and a total one-way delay of 75 ms. q_{ref} is set to 36 packet, and the block error target in the cellular system is 0.1. More details on these simulations are provided in [42, 50].

We compare the following combination of protocols:

- End-to-end TCP Reno.
- End-to-end TCP Reno, using the Eifel at the sending web server, and a Snoop agent at the base station.
- Split-connection using a HTTP proxy. The proxy is located between the operator's network, and the server. There are two TCP Reno connections, one from the web server to the proxy, and another from the proxy to the user's terminal.

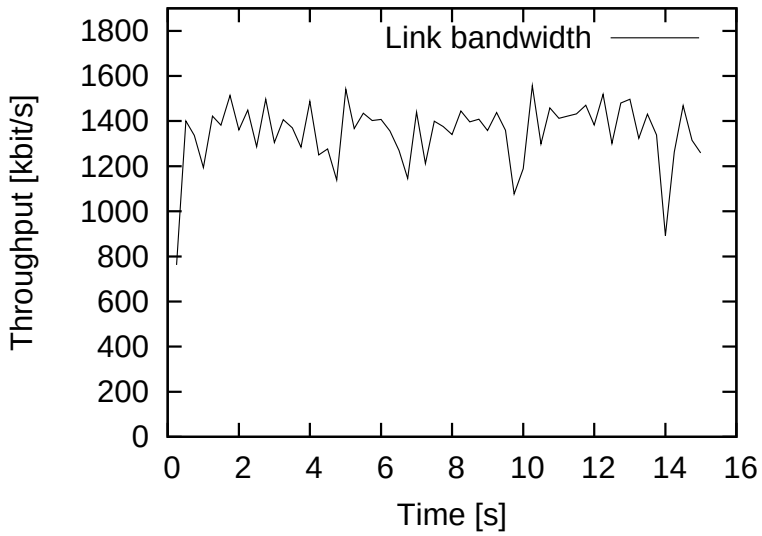

Figure 7.7: The link bandwidth for the channel. Pedestrian A scenario, velocity 3 km/h, distance from base station 450 m.

- Split-connection using the RNF proxy. The connection between web server and proxy uses TCP Reno. The proxy takes advantage of the capacity feedback from the RNC, once every 250 ms, to adjusts its sending window. The receiving terminal need not be aware of this.

- Split-connection with the RNF proxy, combined with Eifel and a Snoop agent at the base station.

In Fig 7.7, we see the time varying capacity of the channel to the user's terminal. In 7.8, we see the resulting throughput for the case of end-to-end TCP Reno, and with a HTTP Proxy. The proxy gives a large gain, which is due to the shorter round-trip time. In this simulation, the one-way delay from the TCP sender to the base station is reduced from to one third, from 75 ms to 25 ms. In Fig. 7.9, we have added the Eifel-protocol at the web-server, and a Snoop agent at the base station. We see a gain in performance, but the throughput is still not close too the available capacity.

The results with the RNF proxy is shown in Fig. 7.10, where it is also compared to a plain HTTP proxy. Finally, in Fig. 7.11, we combine the RNF proxy with Eifel and Snoop.

The average throughput for all the scenarios is summarized in 7.1. We see that using only Eifel and Snoop gives a very modest improvement of the throughput for this channel. This is not so surprising, since both are designed to address problems

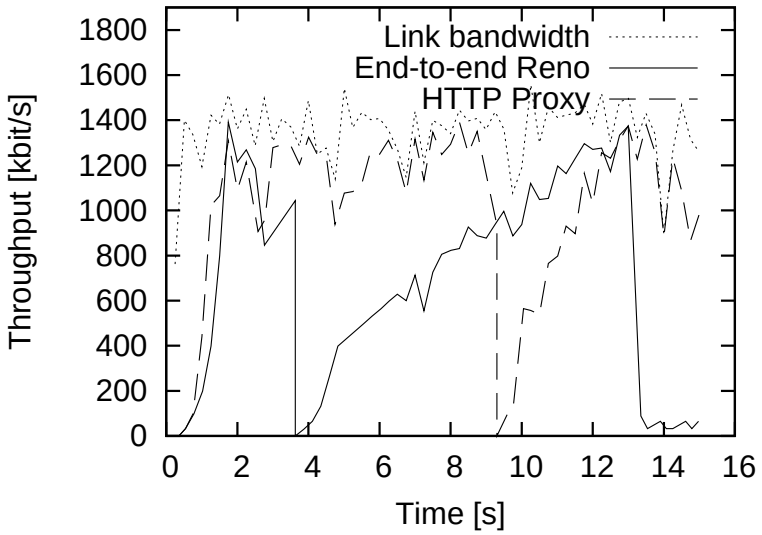

Figure 7.8: Throughput for end-to-end TCP Reno. The figure shows the throughput for both end-to-end Reno, and with an intermediate HTTP proxy.

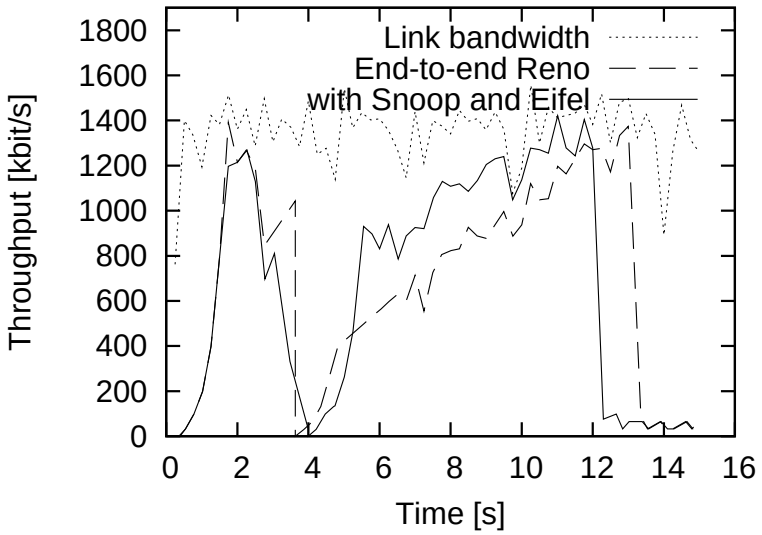

Figure 7.9: Throughput for end-to-end TCP Reno, together with Snoop and Eifel.

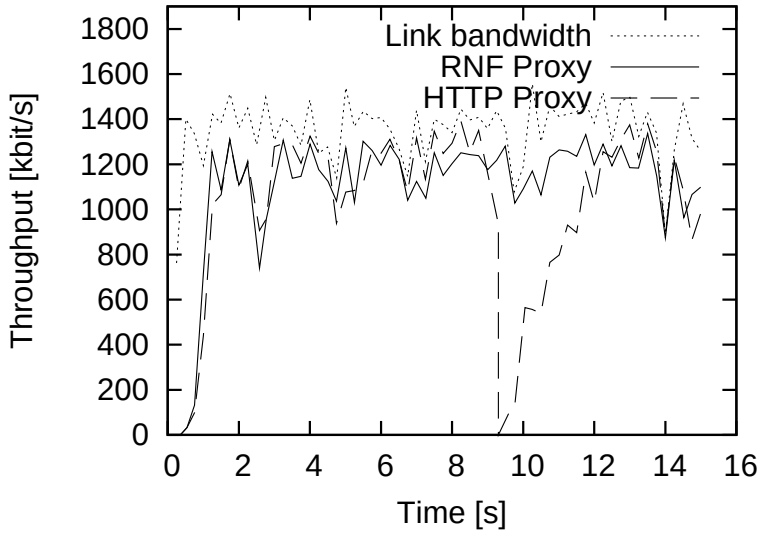

Figure 7.10: Throughput when using the RNF proxy.

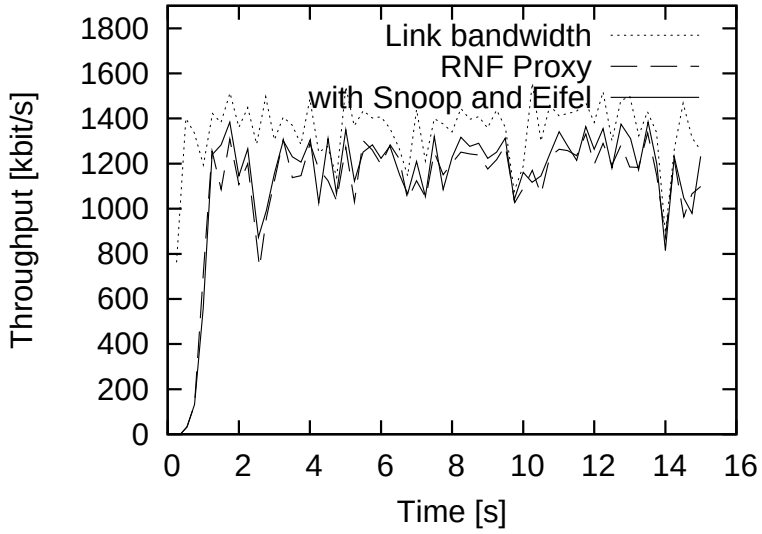

Figure 7.11: Throughput when using the RNF proxy together with Snoop and Eifel.

Scenario	Throughput (kbit/s)	% of capacity
Link bandwidth	1350	100
End-2-end Reno	659	49
with Snoop and Eifel	667	49
HTTP Proxy	1003	74
RNF Proxy	1106	82
with Snoop and Eifel	1137	84

Table 7.1: Average throughput.

with packet losses (Snoop) and outages, which are uncommon due to the link-layer mechanisms. The reduction in round-trip time with split connection, as seen with the HTTP proxy, gives a significant improvement in performance. This is improved further with RNF and, to our surprise, when combining RNF with Snoop and Eifel.

7.6 Tradeoff between throughput and packet delay

One of the objectives of the RNF mechanism is to reduce the user response time, which also corresponds to higher link utilization and fewer queue underflows. An alternative way to increase link utilization is to simply increase the RNC buffer size. This is illustrated in Fig. 7.12, which shows the response time as a function of the queue size, for our second simulation scenario, Sec 7.5, based on radio-layer simulations of HSDPA.

However, when the RNC buffer size is increased, queue delay and jitter also grows, as shown in Fig. 7.13. We see that with TCP Reno, we have a difficult trade-off between high utilization and short packet delay, while the RNF solution achieves high utilization with a fairly small buffer.

A large RNC queue is problematic for several reasons. At hand-over to a different RNC or to a different system, e.g., a Wireless Local Area Network (WLAN), a large RNC queue means that there is more data that has to be transferred or discarded, making a smooth hand-over more difficult. When a user cancels a download, e.g., by clicking the browser stop button and then selecting a different link, all data in the RNC queue will still be transmitted over the radio link, and discarded by the user terminal. In this way, a large RNC queue increases the response time to the user's browsing, and wastes radio resources. Furthermore, higher queueing delay and more jitter implies degraded quality for any real-time traffic sharing the link.

Real-time services provided by the operator, such as voice, need not suffer from a large RNC queueing delay, since the operator is in control of the link layer, and can ensure that this real-time traffic is given priority.

However, no other players have this control. Any third-party real-time service, such as voice, video, or online games, is restricted to end-to-end solutions: Software is installed at the communication end points, but modifications or tweaks to the the network infrastructure are impossible. Therefore, this traffic will most likely share

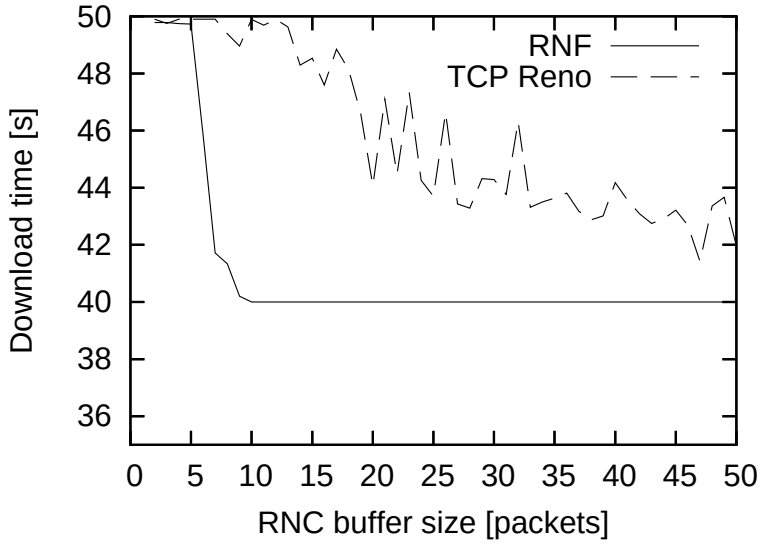

Figure 7.12: Total download time, as a function of RNC buffer size. A download time of 40 s corresponds to almost full utilization of the radio link.

the same RNC queue with web-browsing and downloads.

It is remarkable that by violating the end-to-end principle for one service, web-browsing, we improve the quality for other services which adhere to the end-to-end principle.

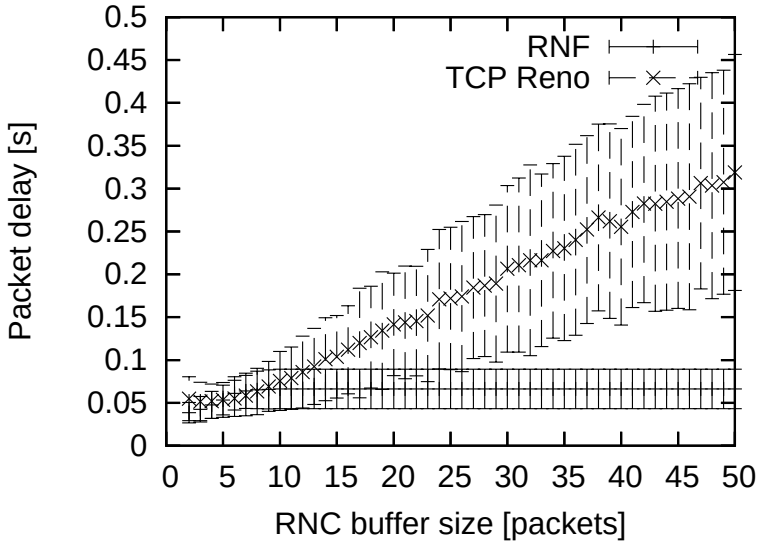

Figure 7.13: Average and standard deviation of the one-way packet delay during the download, as a function of RNC buffer size. With TCP Reno, both average delay and the delay jitter increases with the buffer size.

Chapter 8

Conclusions

Through the chapters of this thesis, we have developed and validated a new model for the window-based packet transmission which is the inner-loop of all window-based congestion control protocols. We have analyzed the properties of this system, and a couple of different outer-loops, in the form of window update rules. Finally, we have designed two new outer-loops, one aimed for general congestion control, and one specialized to cellular systems where some extra cross-layer signalling is possible. So what can we learn about modeling, analysis, and design, of window-based congestion control?

8.1 Modeling

The first lesson is that cross-traffic matters. Consider a bottleneck link shared by congestion controlled traffic and some cross-traffic that is not subject to congestion control. The cross-traffic does not merely reduce the capacity available for the congestion controlled traffic, but the dynamical properties of the congestion control are also affected. For the inner-loop, increasing the cross-traffic increases the static gain between window size and queue size, and it slows down queue convergence.

To give an idea why we have this behavior, let us consider a single fully utilized link and a single flow. Assume the queue is in equilibrium, with a size of 20 packets. We then increase the window size by 10 packets. This results in a burst of packets, and the queue grows instantly to 30 packets. If there is no cross-traffic, this is all that happens: Since the link is already serving the flow at full capacity, there is no change in the stream of ACKs, and no further changes in the sending rate. The queue size of 30 packets is the new equilibrium.

Let us now assume that the link is shared with constant rate cross-traffic, corresponding to one half of the capacity. Before the window change, our flow gets the remaining half. Of the queued packets, 10 belongs to our flow and 10 belongs to the cross-traffic. With the increased window size, the first result is still that the queue grows to 30 packets. But this means that now 2/3 of the queued packets belong

to our flow. Via the ACK-clock, this implies a new higher sending rate, at 2/3 of capacity. Since the cross-traffic keeps sending at half of the capacity, the total traffic exceeds capacity. The queue grows, until it approaches the new equilibrium of 40 packets. At the new equilibrium, our flow and the cross-traffic have 20 queued packets each, and the sending rate of our flow have returned to half of the capacity.

The model developed in Chapter 3 unifies two previous models in the literature: The integrator model, which is accurate for a high level of cross-traffic, and the static model, which is accurate for a low level of cross-traffic. The new model describes the dynamics accurately for any level of cross-traffic, and is validated using `ns2` simulations.

Since the dynamics of the inner-loop, including important properties such as the gain and the time constant, depends on the amount of cross-traffic, it is important to take the cross-traffic into account when designing the window update rules in the outer-loop. We need to construct congestion control so that it either works for arbitrary cross-traffic, or so that it works for some known ranges, e.g., for cross-traffic less than 40% of capacity. The new model enables such analysis.

8.2 Analysis

Fluid-flow modeling of queueing systems results in differential equations with state constraints, because the amount of queued data is necessarily non-negative[1]. These differential equations have discontinuities at the constraint boundary, which violate the Lipschitz condition in standard proofs of existence and uniqueness of solutions. To prove rigorously that such models are well-posed, we have stated and proved existence and uniqueness theorems for differential equations of this form.

Stability results for congestion control protocols are challenging. Ideally, one would like to prove that the system is globally asymptotically stable for arbitrary topologies and arbitrary delays. In our analysis of the inner-loop in window-based congestion control, in Chapter 4, we take a pragmatic approach, and address potential sources of instability separately.

Delay: We have proved that a single link, single source topology is globally stable, for arbitrary delays.

Aggregation: We have proved that a single link, with many sources, each source having its own propagation delay, is locally stable.

Topology: For an arbitrary topology, with no signalling delays, we have proved that the system is globally stable.

[1] Finite buffer sizes are a constraint at the other end of the state space. In the analysis in this thesis, the differential equations are only constrained below. We have either assumed that buffers are unlimited, and aimed for controllers that keep the actual queues small, or used a hybrid model where queue overflow is a discrete event.

While we lack a fully general stability proof, analysis of these three cases give a reasonably indication that the system should be stable under most circumstances. If there is any counter example where the system is unstable, it would probably exhibit some interesting interaction between delay, aggregation and topology.

In the inner-loop stability analysis, we have stability for all values of the parameters. We interpret this as a result of the packet conservation built into the system by Van Jacobson; the total number of packets in the network is bounded by the sum of all the window sizes, and the size of each queue is bounded by the sum of the window sizes of the flows using that queue. To say that window control is stable means that not only are the queues bounded, but they converge to equilibrium values.

For congestion control outer-loops, i.e., when including a window update rule into the model, stability results typically depend on some relation between system parameters and control parameters. For a protocol to be tunable in practice, those relations must be reasonably simple. And to make that true, a very clever analysis may not be sufficient, it also depends on the choice of control and signalling structure, which are design issues. When designing a protocol, it is highly desirable to construct it so that it is easy both to analyze and tune.

8.3 Design

We have presented two congestion control designs, one protocol for general end-to-end congestion control, and one cross-layer mechanism for downloads to a wireless terminal. For both control designs, reduced queueing is the main improvement compared to TCP New Reno. This is essential for the quality of real-time applications sharing the same bottlenecks. It is particularly important in the wireless setting, since the wireless link to the terminal is usually a bottleneck, and it is shared by all applications on the terminal.

To get high quality for all applications, when using a wireless network for both file-sharing and voice calls, like in the emergency response scenario in the introduction (Sec. 1.1), it is necessary to keep queueing delays small.

A new congestion control protocol

When working on the design of the new congestion control protocol in Chapter 6, the guiding principles have been to keep queues small, stay close to TCP New Reno in terms of the other control objectives, including fairness, and to search for control laws that are simple, both from an analysis point of view, and for practical tuning.

The proposed protocol uses the same signalling structure as other TCP/AQM designs: An AQM control in routers, which determines a probability used for marking packets, and a window control at the end points, that reacts to those marks and updates the window size.

For the packet marking in routers, the mark probability p is a very simple continuous function of the instantaneous queue size q, with a single tuning parameter

q_0: $p = q/(q + q_0)$. For the window control part, the control law rule is also fairly simple: Additive increase in the absence of congestion, additive decrease in response to packet marks (which for robustness should be combined with multiplicative decrease in response to packet losses). The only tuning parameter in the window control is the additive increase parameter, usually set to the packet size.

The most striking property of these two control laws is that the system appears to fail gracefully when badly tuned. If the parameter q_0 is too large, the equilibrium queue size grows, but only as $O(\sqrt{q_0})$. And if q_0 is too small, the system becomes unstable, but the simulations indicate that this instability does not lead to any large queue oscillations. Instead, there will be small oscillations close to the queue being empty. The practical effect will be reduced utilization, but not large delays and delay jitter.

The fundamental difference from common TCP/AQM schemes is that the feedback from the network is applied additively to the window sizes. I believe this is a promising approach for making congestion controlled traffic, including peer-to-peer file-sharing of large files, coexist nicely with real-time traffic.

Radio network feedback

In the setting in Chapter 7, we have cross-layer signalling from the Radio Network Controller (RNC) to the proxy which is the TCP sender. The signalling includes information about the capacity of the radio channel, and the queue size in the RNC.

The controller design 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}.

Using `ns2` simulations, and capacity data from detailed radio-layer simulation, the performance of the proposed controller was compared to TCP New Reno. We see improvements in user download time, link utilization, and packet delay. In terms of download time and throughput, the new controller and proxy gives modest improvements, on the order of 10%, when compared to TCP New Reno sender at the same location in the network.

The new controller makes a larger difference when buffering is taken into account. With TCP New Reno, there is a trade-off between throughput and delay, and this trade-off seems to be more difficult for wireless systems like HSDPA with large capacity variations. Throughput is gradually improved when the RNC buffer is made large. A large RNC buffer is problematic for both user response time, and for hand-over. With the new proxy and controller, there is no need for large buffers at the RNC; it is possible to get both high throughput and small delay and delay jitter.

8.4 The future

What the Internet traffic mix will be like in the future is hard to predict. Maybe real-time traffic will remain a small fraction, less than 10%, of the Internet traffic? Or maybe real-time video conferencing (which certainly is more environmentally friendly than international meetings with the corresponding long-distance travelling) will be the next killer application? Or some other unforeseen real-time application that needs much higher capacity than today's voice over IP and online games.

The traffic mix can certainly be of different character in different parts of the Internet. When designing congestion control protocols, it is therefore important to take into account the possibility of a large proportion of the traffic being real-time traffic.

One prediction that it seems quite safe to make, is that we will see more and more devices and users that are connected to the Internet via wireless links. As devices get cheaper, the laptops and mobile phones today, carried by humans, will be joined by wireless sensors for various industrial applications as well as environmental monitoring. The wireless Internet today is a mostly wired backbone, with a single wireless hop from an base station or access point to the wireless terminal. In coming years, we will see some form of multi-hop networks and ad-hoc networks in wide use, e.g., following the roll out of the XO laptops for school children in developing countries.

Will the future Internet use window-based congestion control? I think so. The basic idea, to send a new packet when you know that some previous packet has arrived, is intuitively very sound, and as we have seen in this thesis, it also provides nice stability properties for the network.

Appendices

Appendix A

Mutual information

Assume that p is a stochastic variable in the interval $0 \leq p \leq 1$, with a probability density function f_p. Let y be a mark bit, which is one with probability p, and zero with probability $1-p$. Then the joint probability distribution for p and y is

$$\begin{cases} f_{yp}(0,p) = f_p(p)\,(1-p) \\ f_{yp}(1,p) = f_p(p)\,p \end{cases}$$

The mutual information between y and p is defined by

$$\mathrm{I}(y,p) = \sum_{y\in\{0,1\}} \int_0^1 f_{yp}(y,p)\,\lg \frac{f_{yp}(y,p)}{f_y(y) f_p(p)}\,\mathrm{d}p \tag{A.1}$$

where lg denotes logarithm to base 2. This quantity gives the amount of information we get about p by measuring y (or vice versa).

Proposition A.1 *The mutual information can be expressed as*

$$\mathrm{I}(y,p) = h(\mathrm{E}[p]) - \mathrm{E}[h(p)]$$

where $h(p)$ gives the entropy a binary source with a fixed *probability p,*

$$h(p) = -(p\lg p + (1-p)\lg(1-p))$$

Proof: In the expression for $\mathrm{I}(y,p)$, we need the marginal probability $f_y(y)$. We get

$$f_y(0) = \int_0^1 f_p(p)\,(1-p)\,\mathrm{d}p = 1 - \mathrm{E}[p]$$
$$f_y(1) = \int_0^1 f_p(p)\,p\,\mathrm{d}p = \mathrm{E}[p]$$

Substituting the various probability distribution functions into (A.1) gives

$$\begin{aligned}
\mathrm{I}(y,p) &= \int_0^1 f_p(p)\,(1-p)\lg\frac{f_p(p)\,(1-p)}{(1-\mathrm{E}[p])\,f_p(p)}\,\mathrm{d}p + \int_0^1 f_p(p)\,p\lg\frac{f_p(p)\,p}{\mathrm{E}[p]\,f_p(p)}\,\mathrm{d}p \\
&= \mathrm{E}\left[(1-p)\lg\frac{1-p}{1-\mathrm{E}[p]} + p\lg\frac{p}{\mathrm{E}[p]}\right] \\
&= \mathrm{E}\left[(1-p)\lg(1-p) + p\lg p\right] - \mathrm{E}[1-p]\lg(1-\mathrm{E}[p]) - \mathrm{E}[p]\lg(\mathrm{E}[p]) \\
&= -\mathrm{E}[h(p)] + h(\mathrm{E}[p])
\end{aligned}$$

□

Proposition A.2 *Assume that p is uniformly distributed on the interval $[0, M]$. Then*

$$\mathrm{I}(y,p) = \frac{M}{2} - \left(1-\frac{M}{2}\right)\lg\left(1-\frac{M}{2}\right) - \frac{(1-M)^2}{2M}\lg(1-M) - \frac{1}{2\log 2}$$

In particular, $M = 1$ gives $I(y,p) = 1 - 1/(2\log 2) \approx 0.27$, and M small gives $\mathrm{I}(y,p) = (1/2 - 1/(4\log 2))\,M + O(M^2) \approx 0.14\,M + O(M^2)$.

Proof: The probability distribution for p is

$$f_p(p) = \begin{cases} 1/M & 0 \le p \le M \\ 0 & \text{otherwise} \end{cases}$$

with expected value $E[p] = M/2$. Then

$$\begin{aligned}
\mathrm{E}[p\lg p] &= \frac{1}{M\log 2}\int_0^M p\log p\,\mathrm{d}p \\
&= \frac{M}{2}\lg M - \frac{M}{4\log 2} \\
\mathrm{E}[(1-p)\lg(1-p)] &= \frac{1}{M\log 2}\int_0^M (1-p)\log(1-p)\,\mathrm{d}p \\
&= -\frac{(1-M)^2}{2M}\lg(1-M) + \frac{M-2}{4\log 2}
\end{aligned}$$

where log is the usual logarithm, to base e. Together, we get

$$\begin{aligned}
\mathrm{I}(y,p) &= h(\mathrm{E}[p]) - \mathrm{E}[h(p)] = h(M/2) + \mathrm{E}[p \lg p] + \mathrm{E}[(1-p)\lg(1-p) \\
&= -\frac{M}{2}\lg(M/2) - \left(1-\frac{M}{2}\right)\lg\left(1-\frac{M}{2}\right) \\
&\quad + \frac{M}{2}\lg M - \frac{M}{4\log 2} \\
&\quad - \frac{(1-M)^2}{2M}\lg(1-M) + \frac{M-2}{4\log 2} \\
&= \frac{M}{2} - \left(1-\frac{M}{2}\right)\lg\left(1-\frac{M}{2}\right) - \frac{(1-M)^2}{2M}\lg(1-M) - \frac{1}{2\log 2}
\end{aligned}$$

Putting $M = 1$ gives $I(y,p) = 1/2 - 1/2\lg(1/2) - 1/(2\log 2) = 1 - 1/(2\log 2)$. Taylor expansion around $M = 0$ gives

$$\mathrm{I}(y,p) = \frac{M}{2} + \frac{1}{\log 2}\left\{-\frac{1}{4}M + \frac{1}{24}M^2 + \frac{1}{48}M^3 + O(M^4)\right\}$$

The first non-zero term is $(1/2 - 1/(4\log 2))\,M$. □

Appendix B

Lambert's function

In some of the chapters, we use Lambert's W function [34], defined as the inverse of $z \mapsto ze^z$. The branches are enumerated as $W_k(z)$, where $k \in Z$. There are two real branches. On the real axis, $W_0(x)$ is defined for $x \geq 1/e$, and $W_{-1}(x)$ is defined for $1/e \leq x < 0$. These are illustrated in Fig. B.1.

Equations involving linear and exponential terms can be solved in terms of Lambert's function. E.g., the equation

$$Ax + e^x = B$$

with $A \neq 0$ can be rewritten as

$$\left(\frac{B}{A} - x\right) e^{\left(\frac{B}{A} - x\right)} = \frac{1}{A} e^{\frac{B}{A}}$$

with solutions

$$x = \frac{B}{A} - W_k\left(\frac{1}{A} e^{\frac{B}{A}}\right)$$

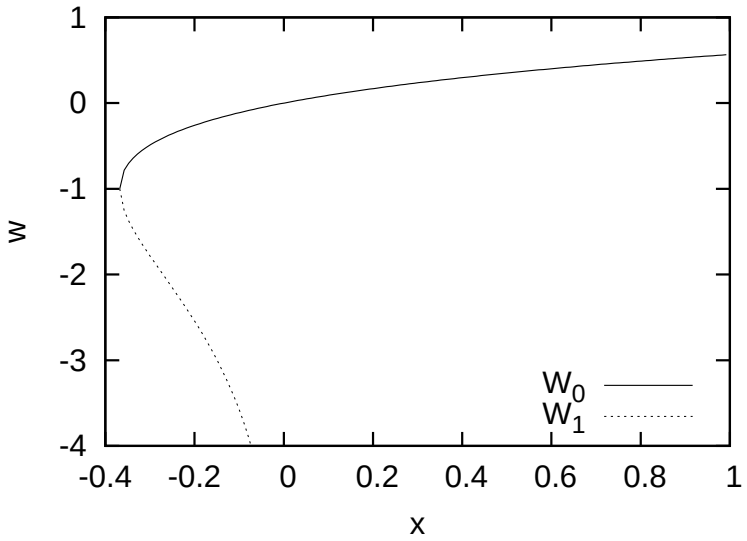

Figure B.1: The two real branches of Lambert's W function. $W_0(x)$ (solid) is defined for all $x \geq -1/e$, while $W_{-1}(x)$ (dotted) is defined on the interval $-1/e \leq x < 0$.

Bibliography

[1] 3rd Generation Partnership Project; Technical specification group radio access network; physical layer procedures (FDD), 2004. TS 25.214 V6.1.0.

[2] 3rd Generation Partnership Project; Technical specification group radio access network; High Speed Downlink Packet Access (HSDPA), overall description, 2006. TS 25.308 V7.0.0.

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

[4] S. Ahmad, I. Awan, and B. Ahmad. Performance modelling of finite capacity queues with complete buffer partitioning scheme for bursty traffic. In *First Asia International Conference on Modelling & Simulation*, pages 264–269, 2007.

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

[6] T. Alpcan and T. Basar. A globally stable adaptive congestion control scheme for Internet-style networks with delay. *IEEE Wireless communications*, 12(6):42–49, December 2005.

[7] E. Altman, C. Barakat, S. Mascolo, N. Möller, and J. Sun. Analysis of TCP Westwood+ in high speed networks. In *International Workshop on Protocols for Fast Long-Distance Networks*, Nara, Japan, January 2006.

[8] Y. H. Aoul, A. Mehaoua, and C. Skianis. A fuzzy logic-based AQM for real-time traffic over Internet. *Computer networks*, 51(16):4617–4633, November 2007.

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

[10] S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin. REM: active queue management. *IEEE Networking*, 15(3):48–53, 2001.

[11] K. Avrachenkov, U. Ayesta, E. Altman, P. Nain, and C. Barakat. The effect of router buffer size on the TCP performance. In *Proceedings of LONIIS workshop*, 2002.

[12] K. Avrachenkov, U. Ayesta, and A. Piunovskiy. Optimal choice of the buffer size in the Internet routers. In *IEEE Conference on Decision and Control and European Control Conference*, Seville, 2005. IEEE CSS.

[13] F. Baccelli and D. Hong. TCP is max-plus linear: and what it tells us on its throughput. In *SIGCOMM*, pages 219–230, 2000.

[14] F. Baccelli and D. Hong. AIMD, fairness and fractal scaling of TCP traffic. In *Proceedings of IEEE INFOCOM*, pages 229–238, New York, June 2002.

[15] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. Improving TCP/IP performance over wireless networks. In *Proc. 1st ACM Conference on Mobile Communications and Networking (Mobicom)*. ACM, Berkeley 1995.

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

[17] 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.

[18] E. Blanton, M. Allman, K. Fall, and L. Wang. A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP. RFC 3517, April 2003.

[19] S. Bohacek, J. P. Hespanha, J. Lee, and K. Obraczka. Analysis of a TCP hybrid model. In *Annual Allerton Conference on Communication, Control, and Computing*, 2001. Extended version available at `http://www.ece.ucsb.edu/~hespanha/techrep.html`.

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

[21] Allan Borodin, Jon Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queueing theory. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 376–385, 1996.

[22] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. LNCS. Springer, 2001.

[23] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[24] L. S. Brakmo and L. L. Peterson. TCP Vegas: end-to-end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, 1995.

[25] B. Briscoe. Flow rate fairness: Dismantling a religion. *Computer communication review*, 37(2):65–74, April 2007.

[26] R. Bush and D. Meyer. Some Internet architectural guidelines and philosophy. RFC 3439, December 2002.

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

[28] 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.

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

[30] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, January 2007.

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

[32] J. Y. Choi, K. Koo, J. S. Lee, and S. H. Low. Global finite-time convergence of TCP vegas without feedback information delay. *International journal of control automation and systems*, 5(1):70–78, February 2007.

[33] C. Chrysostomou, A. Pitsillidesa, L. Rossidesa, M. Polycarpoub, and A. Sekerciogluc. Congestion control in differentiated services networks using fuzzy-RED. *Control Engineering Practice*, 11(10):1153–1170, October 2003.

[34] R. M. Corless, G. H. Gonnet, D. E. H. Hare, D. J. Jeffrey, and D. E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996.

[35] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 1991.

[36] Rene L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[37] Rene L. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[38] A. Dhamdhere, H. Jiang, and C. Dovrolis. Buffer sizing for congested Internet links. In *Proceedings of IEEE INFOCOM*, 2005.

[39] J. Postel (ed.). Transmission control protocol. RFC 793, September 1981.

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

[41] A. K. Erlang. Telefon-ventetider. *Matematisk Tidskrift*, B:25–42, 1920. Digitized at `http://runeberg.org/matetids/1920b/0029.html`.

[42] Marco Fiorenze. Downlink TCP proxy solutions for interactive gaming over HSDPA and 3G networks. Master's thesis, KTH, September 2007.

[43] S. Floyd. HighSpeed TCP for large congestion windows. RFC 3649, December 2003.

[44] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. `http://www.icir.org/floyd/papers/adaptiveRed.pdf`, August 2001.

[45] S. Floyd, T. Henderson, and A. Gurtov. The NewReno modification to TCP's fast recovery algorithm. RFC 3782, April 2004.

[46] 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.

[47] D. Gamarnik and D. Katz. On deciding stability of multiclass queueing networks under buffer priority scheduling policies. eprint arXiv:0708.1034v1, November 2007. `http://arxiv.org/pdf/0708.1034`.

[48] Y. Ganjali and N. McKeown. Update on buffer sizing in Internet routers. *Computer Communication Review*, 36(5):67–70, October 2006.

[49] T. Gillespie. Engineering a principle: 'End-to-end' in the design of the Internet. *Social Studies of Science*, 36(3):427–457, June 2006.

[50] Daniele Girella. Downlink TCP proxy solutions over HSDPA with flow aggregation and user behavior. Master's thesis, KTH, September 2007.

[51] S. Gorinsky, A. Kantawala, and J. Turner. Link buffer sizing: A new look at the old problem. In *Proceedings of IEEE Symposium on Computers and Communications*, pages 507–514, June 2005.

[52] L. A. Grieco and S. Mascolo. Performance evaluation and comparison of Westwood+, New Reno and Vegas TCP congestion control. *ACM Computer Communication Review*, 34(2), April 2004.

[53] L. Guan, M. E. Woodward I. U. Awan and, and X. Wang. Discrete-time performance analysis of a congestion control mechanism based on RED under multi-class bursty and correlated traffic. *Journal of Systems and Software*, 80(10):1716–1725, October 2007.

[54] Gerhard Haßlinger, Joachim Mende, Rüdiger Geib, Thomas Beckhaus, and Franz Hartleb. Measurement and characteristics of aggregated traffic in broadband access networks. In *Proceedings of Intnernational Teletraffic Congress: Managing Traffic Performance in Converged Networks*, volume 4516 of *LNCS*, pages 998–1010, Ottawa, June 2007. Springer.

[55] J. P. Hespanha, S. Bohacek, K. Obraczka, and J. Lee. Hybrid modeling of TCP congestion control. In M. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 291–304. Springer-Verlag, Berlin, Germany, 2001.

[56] C. Hollot, V. Mistra, D. Towsley, and W. B. Gong. A control theoretic analysis of RED. In *IEEE Infocom 2001*, 2001.

[57] C. V. Hollot, V. Misra, W.-B. Gong, and D. Towsley. On designing improved controllers for AQM routers supporting TCP flows. In *IEEE Infocom*, pages 1726–1734, April 2001.

[58] 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.

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

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

[61] K. Jacobsson, H. Hjalmarsson, and K. H. Johansson. Unbiased bandwidth estimation in communication protocols. In *Proc. of the 16th IFAC World Congress*, Prague, July 2005.

[62] R. S. Jayaram and I. Rhee. A case for delay-based flow control in CDMA 2.5G networks. In *International Conference on Ubiquitous Computing*, Washington, 2003.

[63] D. Katabi, M. Handley, and C. Rohrs. Internet congestion control for future high bandwidth-delay product environments. In *ACM Sigcomm*, August 2002.

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

[65] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of Operational Research Society*, 49:237–252, 1998.

[66] T. Kelly. Scalable TCP: Improving performance in highspeed wide area networks. *Computer Communication Review*, 32(2), 2003.

[67] H. K. Khalil. *Nonlinear systems*. Prentice Hall, 2nd edition, 1996.

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

[69] R. King., R. Baraniuk, and R. Riedi. TCP-Africa: an adaptive and fair rapid increase rule for scalable TCP. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1838–1848, March 2005.

[70] 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.

[71] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Minimizing the overhead in implementing flow-aware networking. In *Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems*, pages 153–162, Princeton, 2005. ACM.

[72] R. J. La and P. Ranjan. Asymptotic stability of a rate control system with communication delays. *IEEE Transactions on automatic control*, 52(10):1920–1925, October 2007.

[73] L. Le, J. Aikat, K. Jeffay, and F. D. Smith. The effects of active queue management on web performance. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 265–276, Karlsruhe, 2003. ACM.

[74] Junsoo Lee, Stephan Bohacek, ao P. Hespanha Jo and Katia Obraczka. Modeling communication networks with hybrid systems. *IEEE/ACM Trans. Netw.*, 15(3):630–643, 2007.

[75] Kaarina Lehtisalo. *The History of NORDUnet—Twenty-Five Years of Networking Cooperation in the Nordic Countries*. NORDUnet, 2005. `http://www.nordu.net/history/index.html`.

[76] A. Leizarowitz, R. Stanojevic, and R. Shorten. Tools for the analysis and design of communication networks with markovian dynamics. *Proceedings of IEE, Control Theory and Applications*, 153(5):506–519, Sept 2006.

[77] I. C. Lestas and G. Vinnicombe. Scalable decentralized robust stability certificates for networks of interconnected heterogeneous dynamical systems. *IEEE Transactions on automatic control*, 51(10):1613–1625, October 2006.

[78] S. Liu, sar T. Ba and R. Srikant. TCP-Illinois: A loss and delay-based congestion control algorithm for high-speed networks. In *Proc. First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, Pisa, October 2006.

[79] S. H. Low and D. E. Lapsley. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, December 1999.

[80] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, pages 28–43, February 2002.

[81] S. H. Low, L. Peterson, and L. Wang. Understanding Vegas: a duality model. *Journal of ACM*, 49(2):207–235, 2002.

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

[83] S. Manfredi, M. Bernardo, and F. Garofaloa. Reduction-based robust active queue management control. *Control Engineering Practice*, 15(2):177–186, February 2007.

[84] Ian Marsh, Fengyi Li, and Gunnar Karlsson. Wide area measurements of VoIP quality. In *Quality of Future Internet Services*, Stockholm, Sweden, October 2003.

[85] N. C. Martins, M. A. Dahleh, and J. C. Doyle. Fundamental limitations of disturbance attenuation in the presence of side information. *IEEE Transactions on Automatic Control*, 52(1):56–66, January 2007.

[86] 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.

[87] S. Mascolo and F. Vacirca. Congestion control and sizing router buffers in the Internet. In *IEEE Conference on Decision and Control and European Control Conference*, pages 6750–6755, Seville, 2005. IEEE CSS.

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

[89] J. Mo and J. Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking*, 8(5):556–567, October 2000.

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

[91] N. Möller, K. H. Johansson, and H. Hjalmarsson. Making retransmission delays in wireless links friendlier to TCP. In *IEEE Conference on Decision and Control*. IEEE, 2004.

[92] J. Nagle. Congestion control in IP/TCP Internetworks. RFC 896, January 1984.

[93] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers. RFC 2474, December 1998.

[94] *The Network Simulator ns-2*. Information Sciences Institute, University of Southern California. `http://www.isi.edu/nsnam/ns`.

[95] F. Paganini, J. Doyle, and S. Low. Scalable laws for stable network congestion control. In *IEEE CDC*, Orlando, FL, 2001.

[96] M. Peet and S. Lall. Global stability analysis of a nonlinear model of Internet congestion control with delay. *IEEE Transactions of automatic control*, 52(3):553–559, March 2007.

[97] J. Postel. User datagram protocol. RFC 768, August 1980.

[98] Eurane project. *Enhanced UMTS Radio Access Network Extensions for ns-2*. `http://www.ti-wmc.nl/eurane/`.

[99] Wubi Qin and Qian Wang. An LPV approximation for admission control of an internet web server: Identification and control. *Control Engineering Practice*, 15(12):1457–1467, December 2007.

[100] V. Raghunathan and P. R. Kumar. A counterexample in congestion control of wireless networks. *Performance evaluation*, 64(5):399–418, June 2007.

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

[102] I. Rhee and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. In *International Workshop on Protocols for Fast Long-Distance Networks*, Lyon, February 2005.

[103] J. W. Roberts. Traffic theory and the Internet. *IEEE Communications Magazine*, January 2001.

[104] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, January 2001.

[105] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on computer systems*, 2(4):277–288, November 1984.

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

[107] Bruce Schneier. Crypto-gram newsletter, May 2005. `http://www.schneier.com/crypto-gram-0005.html`.

[108] R. Shorten, F. Wirth, and D. Leith. A positive systems model of TCP-like congestion control: Asymptotic results. *IEEE-ACM Transactions on networking*, 14(3):616–629, June 2006.

[109] R. Srikant and S. Deb. Rate-based versus queue-based models of congestion control. *IEEE Transactions on automatic control*, 51(4):606–619, April 2006.

[110] J. Sun and M. Zukerman. RaQ: A robust active queue management scheme based on rate and queue length. *Computer Communications*, 30(8):1731–1741, June 2007.

[111] J. Sun, M. Zukerman, and M. Palaniswami. Stabilizing RED using a fuzzy controller. In *IEEE International Conference on Communications*, pages 266–271, Glasgow, June 2007. IEEE.

[112] K. Tan, Jingmin Song, Qian Zhang, and Murari Sridharan. A compound TCP approach for high-speed and long distance networks. In *Proceedings of IEEE INFOCOM*, Barcelona, April 2006.

[113] M. Vidyasagar. *Nonlinear systems analysis*. Prentice-Hall, second edition, 1993.

[114] J. Wang, D. X. Wei, and S. H. Low. Modelling and stability of FAST TCP. In *Proceedings of IEEE Infocom*, Miami, March 2005.

[115] D. X. Wei, C. Jin, S. H. Low, and S. Hedge. FAST TCP: Motivation, architecture, algorithms, performance. *IEEE-ACM Transactions on networking*, 14(6):1246–1259, December 2006.

[116] J. Wroclawski. The use of RSVP with IETF integrated services. RFC 2210, September 1997.

[117] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *IEEE Transactions on networking*, 15(1):133–144, February 2007.

[118] F. Yin, G. M. Dimirovski, and Y. Jing. Robust stabilization of uncertain input delay for Internet congestion control. In *American Control Conference*, June 2006.

[119] L. Ying, G. E. Dullerud, and R. Srikant. Global stability of Internet congestion controllers with heterogeneous delays. *IEEE-ACM Transactions on networking*, 14(3):579–591, June 2006.

[120] F. Zheng and J. Nelson. An H_∞ approach to congestion control design for AQM routers supporting TCP flows in wireless access networks. *Computer Networks*, 51(6):1684–1704, April 2007.

[121] X. Zhou, J. Wei, and C.-Z. Xu. Quality-of-service differentiation on the Internet: A taxonomy. *Journal of Network and Computer Applications*, 30(1):354–383, January 2007.