Uitnodiging

Op dinsdag 14 maart 2006 zal ik mijn proefschrift "Application-oriented Link Adaptation for IEEE 802.11" en bijbehorende stellingen verdedigen.

De plechtigheid vangt aan om 13.00 uur in de Senaatszaal van de Aula van de Technische Universiteit Delft. Voorafgaand, om 12.30 uur, zal ik een korte presentatie over mijn promotieonderzoek geven. Aansluitend aan de promotie is er een receptie in de Aula.

U bent van harte uitgenodigd voor deze promotie en receptie.

On Tuesday, March 14th 2006, I will defend my Ph.D. thesis "Application-oriented Link Adaptation for IEEE 802.11" and the accompanying propositions.

The ceremony will take place at 13:00 in the Senaatszaal of the Aula of the Delft University of Technology. Before that, at 12:30 there will be a short presentation of my research. After the defense, a reception will take place.

You are welcome to both the defense and the reception.

Ivaylo Haratcherev

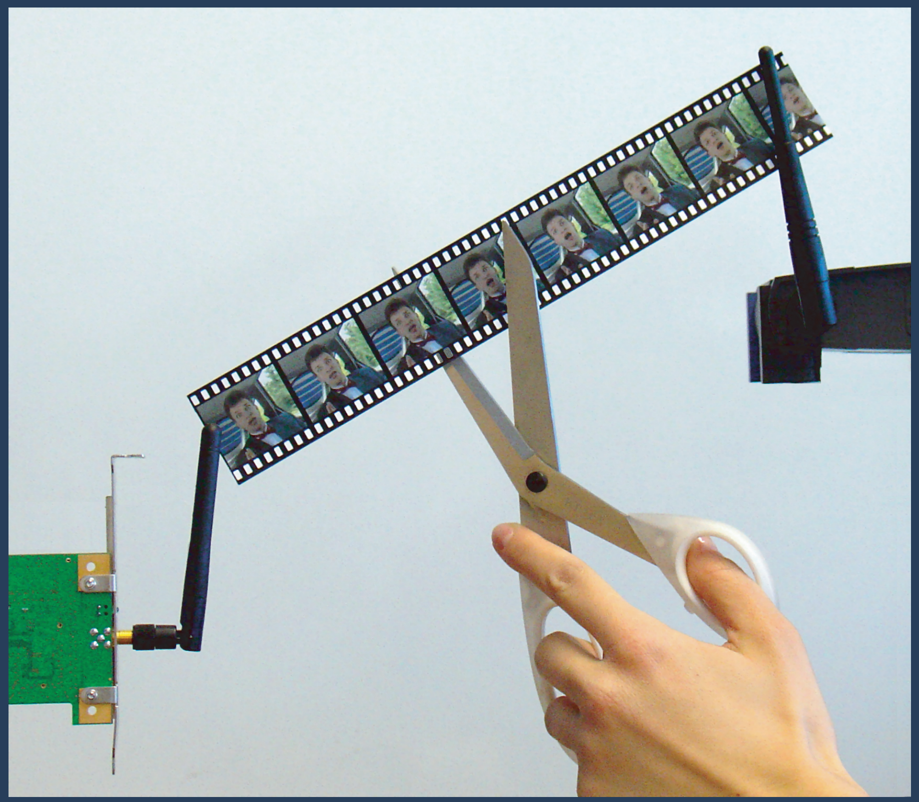
ISBN-10: 90-9020513-6
ISBN-13: 978-90-9020513-7

Application-oriented Link Adaptation for IEEE 802.11

Ivaylo Haratcherev

Application-oriented Link Adaptation for IEEE 802.11

Ivaylo Haratcherev



Delft, 2006

Stellingen

behorende bij het proefschrift

Application-oriented Link Adaptation for IEEE 802.11

Ivaylo Haratcherev

1. Zelfs de kleinste verbetering aan het (Radio) Link Adaptatie algoritme op de fysieke laag levert een veel grotere winst op voor streaming applicaties dan welke verandering dan ook aan enig andere laag van de netwerk protocol stack.
   [dit proefschrift, Hoofdstuk 3 en 4]
2. Objectieve kwaliteitsnormen zoals PSNR komen niet overeen met de menselijke waardering van video beelden. Deze discrepantie is het sterkst bij link drop-outs.
   [dit proefschrift, Hoofdstuk 3]
3. Alhoewel een Kalman filter een effectieve techniek is om met verstoorde signalen om te gaan, is het geen geen optie om deze techniek te gebruiken in de Link Adaptatie van een 802.11 draadloos netwerk.
   [dit proefschrift, Hoofdstuk 2]
4. Een magnetron kan zeer goed gebruikt worden als radio stap responsie generator bij het onderzoek naar draadloze netwerken. In die hoedanigheid presteert het zelfs beter dan gespecialiseerde apparatuur, zowel qua prijs als qua effectiviteit.
   [dit proefschrift, Hoofdstuk 3]
5. Nieuwe technologie heeft de grootste kans van slagen indien de hoeveelheid tijd en geld die gespendeerd wordt aan de standaardisatie en achterwaartse compatibiliteit van gelijke orde is als de hoeveelheid gespendeerd aan de ontwikkeling van deze technologie.
6. Wetenschappelijk onderzoek concentreert zich meestal op de werking van een systeem in een bepaalde, wel omschreven context. Het is echter noodzakelijk juist de overgangen *tussen* verschillende contexten te bestuderen en rekening te houden met allerlei randgevallen en uitzonderingen.
7. De ultieme vorm van persoonlijk draadloze communicatie zal een in het hoofd geïmplanteerde radio chip worden.
8. In plaats van geavanceerde systemen te ontwerpen die alleen functioneren als alle componenten foutloos werken, zouden onderzoekers moeder natuur moeten volgen en zich richten op de ontwikkeling van systemen die ook functioneren als een aanzienlijke fractie van de componenten stuk is.
9. De mensheid zal altijd oplossingen vinden voor grote, reeds lang voorziene problemen, maar alleen als het vijf voor twaalf is.
10. Een belangrijk gegeven in de Nederlandse bestuurscultuur is dat elke regelgeving die functioneert onmiddellijk veranderd dient te worden.

Deze stellingen worden opponeerbaar en verdedigbaar geacht en zijn als zodanig goedgekeurd door de promotoren, Prof. dr. ir. H.J. Sips en Prof. dr. ir. R.L. Lagendijk.

Propositions

accompanying the thesis

Application-oriented Link Adaptation for IEEE 802.11

Ivaylo Haratcherev

1. Even the smallest effort spent on improving the performance of (Radio) Link Adaptation algorithms is much more beneficial for improving performance of streaming applications than any effort done at other network layers.
   [this thesis, Chapters 3 and 4]
2. When dealing with link drop-outs, objective criteria for evaluating video quality, like PSNR, are inconsistent with the human perception of quality.
   [this thesis, Chapter 3]
3. Although Kalman filtering is an effective technique for dealing with noisy signals, it is not suitable for use in an 802.11 Link Adaptation controller.
   [this thesis, Chapter 2]
4. A microwave oven makes a good radio step response generator, and beats in performance and price specialized radio chamber equipment.
   [this thesis, Chapter 3]
5. The best chance for success of a new technology is when the effort spent on standardization and backwards compatibility is of the same order of magnitude as the effort spent on developing the technology.
6. Instead of centering research efforts around optimizing a system within specific modes of operation, more attention should be paid to how and when a system switches between modes, and how to deal with border conditions.
7. The ultimate in personal wireless communications will be a radio chip implanted in your head.
8. Researchers should follow nature's example and move their focus from systems that need close to 100% of their components to function properly, to systems that do their job even when a significant number of their components are out of order.
9. Humanity will find solutions for long foreseen problems that have important consequences, but not until the very last moment.
10. A main rule of any big organization, like Dutch administration, is: If something works – change it!

These propositions are considered opposable and defendable and as such have been approved by the supervisors, Prof. dr. ir. H.J. Sips and Prof. dr. ir. R.L. Lagendijk.

Application-oriented Link Adaptation for IEEE 802.11

Application-oriented Link Adaptation for IEEE 802.11

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. dr. ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen
op dinsdag 14 maart 2006 om 13.00 uur

door

Ivaylo Jivkov HARATCHEREV

Systems and Control Engineer
Technical University of Sofia
geboren te Sofia, Bulgarije.

Dit proefschrift is goedgekeurd door de promotoren:

Prof. dr. ir. H.J. Sips
Prof. dr. ir. R.L. Lagendijk

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof. dr. ir. H.J. Sips	Technische Universiteit Delft, promotor
Prof. dr. ir. R.L. Lagendijk	Technische Universiteit Delft, promotor
Dr. K. G. Langendoen	Technische Universiteit Delft, toegevoegd promotor
Prof. dr. ir. I.G.M.M. Niemegeers	Technische Universiteit Delft
Prof. dr. R. Babuska	Technische Universiteit Delft
Dr. ir. C.Th.A.M. de Laat	Universiteit van Amsterdam
Prof. dr. H. Karl	Universität Paderborn

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 122.

Email: `I.Haratcherev@ewi.tudelft.nl`

ISBN-10: 90-9020513-6
ISBN-13: 978-90-9020513-7

Printed in the Netherlands.

Acknowledgements

GETTING a Ph.D. research to a successful end is not a trivial mission. On the path to completion there were people that helped me, and there were others that were standing on the way. I want to thank both groups. The first – for obvious reasons. The second – because “Tougher the battle, sweeter the victory”.

I would like to thank my supervisor Koen Langendoen for his guidance and relaxed attitude, which helped me a lot to get through this big task easier. Also, lots of thanks to my promotors, Henk Sips and Inald Lagendijk. I am grateful to Prof. R. Babuska for his valuable comments on this thesis. Special thanks go to Jacco for the nice collaboration we had together.

I would like to thank my mother and my sister for their support.

Благодаря на всички приятели от Българската маса за незабравимите (весели и неописуеми) моменти заедно, както и на всички в България които ми оказваха духовна подкрепа.

I would like to thank two of my colleagues (roommates) as well: Ana – for providing a wonderful working atmosphere; and Johan – for the great ideas we realized together.

Delft,
March 2006

Ivaylo Haratcherev

Contents

List of abbreviations and acronyms

ACK	acknowledgment
AP	access point
BER	bit error rate
BPSK	binary phase shift keying
BSS	basic service set
CA	collision avoidance
CBR	constant bit rate
CCK	complementary code keying
CD	collision detection
CDMA	code division multiple access
CS	carrier sense
CSI	channel state information
CSMA	carrier sense multiple access
CTS	clear to send
CW	contention window
DBPSK	differential binary phase shift keying
DCF	distributed coordination function
DIFS	distributed (coordination function) interframe space
DQPSK	differential quadrature phase shift keying
DSSS	direct sequence spread spectrum
EIFS	extended interframe space
EIRP	equivalent (effective) isotropic radiated power
EMI	electromagnetic interference
FDMA	frequency division multiple access
FER	frame error rate
FHSS	frequency-hopping spread spectrum

HSDPA	high-speed downlink packet access
IFS	interframe space
IR	infrared
ISM	industrial, scientific, and medical
LA	link adaptation
LAN	local area network
MAC	medium access control
MIMO	multiple input / multiple output
MPDU	MAC protocol data unit
MPLS	multiprotocol label switching
MSDU	MAC service data unit
NAV	network allocation vector
NIC	Network Interface Card
OFDM	orthogonal frequency division multiplexing
PBCC	packet binary convolutional code
PCF	point coordination function
PER	packet error rate
PHY	physical (layer)
PLCP	physical layer convergence protocol
PLR	packet loss rate
PN	pseudo-noise (code sequence)
PSNR	peak signal-to-noise ratio
QAM	quadrature amplitude modulation
QoS	quality of service
QPSK	quadrature phase shift keying
RF	radio frequency
RSCD	rapid SSI change detector
RSSI	received signal strength indication
RTS	request to send
RX	receive or receiver
SIFS	short interframe space
SNR	signal-to-noise ratio
SRF	Step Response Function
SSI	signal strength indication
SSIA	signal strength indication of the acknowledged frames
STA	station
STAC	SSI thresholds adaptation circuit
TDMA	time division multiple access

TX	transmit or transmitter
UMTS	Universal Mobile Telecommunications System
U-NII	Unlicensed National Information Infrastructure
VoIP	Voice over IP
VRCA	video rate-control algorithm
WEP	wired equivalent privacy
WPA	Wi-Fi Protected Access

Chapter 1

Introduction

THE first publicly available mobile communication services date back to the 1940s. These services started with bulky expensive analog devices that provided low quality and insufficient capacity (a few people who can talk simultaneously). It was not until the late 80's and the 90's of the 20th century when wireless communications became really accessible to the mass public. This became possible when the underlying technology shifted to digital and cellular. Together with the development of the voice wireless networks, and catching up with the evolution of the Internet, wireless data network devices followed.

One of the first popular wireless products was the pre-802.11 standard WaveLAN, which offered only 2 Mbit/s [Cla94]. WaveLAN went through various modifications that brought it to a de-facto 802.11 product, and eventually turned into the well-known (and faster) 802.11b Orinoco PCMCIA card (see Figure 1.1). Although 802.11b and 802.11a, which were both aimed at improving the data throughput, were published in the same year (1999), the first commercial 802.11a chipset – the AR5000 made by Atheros – appeared as late as 2001. Soon the first cards based on this chipset followed (Figure 1.2). In our work we used Proxim cards extensively for testing and evaluation of various algorithms.

Now, in 2006, the 802.11 products are even smaller (like the one in Figure 1.3), and very often completely integrated in notebooks, PDAs, cameras, Internet phones and so on. Despite all these improvements and the overall speed increase (the 54Mbps of 802.11g and 802.11a, versus the 11Mbps of 802.11b) still very often users are not satisfied with the quality of service these devices offer. This holds especially for interactive applications with real-time constraints like multimedia streaming.

1.1 The problem of wireless links

The root of the problem that we address in this thesis lies in the very nature of the wireless link. A wireless link is extremely vulnerable and sensitive to

Figure 1.1: *The popular Orinoco 802.11b card made by Lucent [1999]*

Figure 1.2: *One of the first 802.11a cards - by Proxim [2002]*

Figure 1.3: *Typical modern 802.11b/g USB card [2005]*

all kinds of disturbances and effects, and this results in highly dynamic and unstable link quality characteristics. Examples of effects influencing the wireless link are variations in distance between stations, objects obstructing the way of radio waves, and reflections of the radio signal, interfering transmissions from other stations.

The result is well-known to everyone. A user usually gets frustrated by the drop-outs in the conversation with the other party, or by the bad, freezing video that she/he gets while watching her/his favourite singer's new clip. The reason for the stalling video is the excessive packet loss and latency problems that exist in an unstable wireless link. Streaming applications like video are very sensitive to such problems.

These problems cannot be tackled by over-provisioning, as it is done in the wired Internet. There are two reasons for that. The first is that there are unsuccessful transmissions due to interference. The latter is a result of the radio medium being a shared resource. Unsuccessful transmissions cause increased packet losses and packet delays because of the need that the lost data is retransmitted again. The second reason is that in the world of radio only a limited amount of power can be used for radio transmissions. This results in limited range and unsuccessful transmission attempts due to weak signal. The power limitation is a consequence of the fact that mobile devices normally do not have sufficient energy resources, and that increased radio power will cause more interference to others using the same medium.

The standard solution to the problem of radio link instability is called Link Adaptation (LA), and is applied in many existing wireless communication systems, such as GSM for example [Han97; Que99]. LA is the process of adjusting the wireless link parameters, following the changes of the radio channel conditions so that an optimal link quality is achieved. The controlled link parameters vary from system to system; examples for such parameters are transmission radio power, modulation/coding scheme type, and channel/frequency. The channel conditions that mostly influence the quality of the link are pathloss (the radio wave propagation losses occurring between the transmitter and the receiver) and interference. The latter could be Rayleigh fading caused by multipath (propagation phenomenon resulting in radio waves reaching the receiver's antenna by two or more paths), or could be caused by signals from other stations. Since usually for the systems that need to employ LA the channel conditions changes are frequent and rapid, LA can be a very dynamic process. For example, in HSDPA (High-Speed Downlink Packet Access) and in UMTS (Universal Mobile Telecommunications System) – 3.5G and 3G mobile phone standards – LA takes place every 2 ms [Wik06].

Unfortunately, Link Adaptation is neither part of the 802.11 standard, nor of any subsequent 802.11 amendment. The reason that no significant attention has been paid to LA issues is twofold:

- The first reason is due to the complexity of LA and the multidisciplinary kind of knowledge that it requires to be done properly. Since the problems of LA were left out of the 802.11 standard, the people that had to deal with this control were normally researchers and developers within radio chipsets manufacturing companies. These scientists concentrated mainly on the stability issues of LA algorithms and on the throughput performance – both very important issues for a wide commercial deployment. Therefore packet loss and packet latency issues were overlooked, which was not a problem for download applications that by nature are quite tolerant to packet loss and latency. Packet losses are taken care of by TCP, and latency does not matter as long as the mean throughput is high enough.

- The second reason is the low penetration so far of streaming applications in the life of today's average Internet user. The low popularity of streaming applications fuels the negligence about packet loss and latency. This reluctance to use the praised Internet (video)phones and alike is due to the overall lack of quality mechanisms in Internet, which despite the long QoS-related efforts still does not get any better than a best-effort service.

We believe that soon real-time streaming applications (VoIP, video-telephony, etc.) will gain their long awaited popularity and spread. Therefore, those applications are the ones that are addressed by the research presented here.

Another belief is that changing or amending a standard is a step normally only taken when there is a very strong demand for it. Such a demand cannot come from a niche application, which currently real-time streaming is. Furthermore, real-time multimedia will not get a big boost until quality problems in the data networks are solved end-to-end. So, to help streaming applications break that vicious circle they are trapped into, the research presented in this thesis aims at clearing the way by attacking the last bastion of bad QoS - the wireless - without any changes to be made to the 802.11 standard.

1.2 Approach

We take a two-stage approach to the problems of the wireless link. The first stage is obvious - apply a better, i.e. more responsive, Link Adaptation method. In this thesis we use a combination of an existing stable algorithm (suited for download applications) and a rapid link quality feedback to produce a novel rate controller that adapts to changes very fast, while still producing stable performance. Prompt adaptation reduces packet loss and latency and helps multimedia applications to deliver their content in a timely fashion.

The second stage is to notify everyone how the radio link is doing right now. In other words, inform the application how the link is performing. In that way the application can adjust its demands accordingly and follow the changes in the provided data transport quality, to avoid the annoying break-downs in a multimedia stream.

The results that we achieve by applying both techniques are very good and the reader can get a feeling about them by looking at Figure 1.4. On the left is a typical case of a stalled video while transmitting over an 802.11 connection employing a standard Link Adaptation algorithm. On the right is the same video over the same connection and under the same link conditions, but this time using our combination of an advanced hybrid Link Adaptation method with cross-layer signalling (i.e. exchange of control information between network layers).

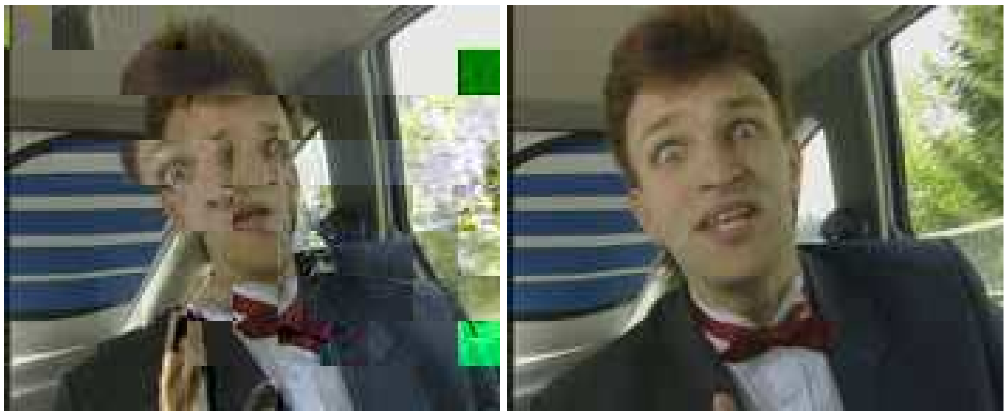


Figure 1.4: *Standard Link Adaptation (left) and hybrid Link Adaptation with cross-layer signalling (right).*

1.3 Contributions

In this thesis two approaches are presented to improve the application-level network quality in wireless connections, and consequently, the performance of streaming applications in particular. The first, and most important contribution is a novel hybrid Link Adaptation algorithm that significantly reduces packet losses and delays typical for standard Link Adaptation algorithms. Our algorithm also behaves better in a most likely scenario that the channel is shared with other users.

The second contribution is the specific cross-layer signalling scheme in which the link layer provides current and predicted throughput feedback to the application layer. This feedback is then used by the application (video-encoder) to adjust its generated data rate accordingly and in timely fashion. The benefits of this adaptation are that the application has the time and knowledge to react to changes of the link quality and therefore can avoid hiccups in the multimedia streaming process – the kind of problems users are most sensitive to.

To facilitate the standardization of cross-layer communications we developed RA*I*L – the Radio Abstraction Information Layer. It is an application programming interface (API) that should aid both the developers of wireless card drivers and the developers of streaming applications. RA*I*L is built on the widely-distributed Linux Wireless Extensions by Jean Tourrilhes [Jea96] to ease the transition of existing applications/drivers and to promote consistency.

In our work we have chosen the path of real implementation as a research method, as opposed to simulations. Although requiring a lot of efforts, such an approach is extremely beneficial in terms of validating the performance improvements that come as result of our research, and in terms of testing our algorithms in real scenarios.

1.4 Thesis outline

The logic of organization this thesis follows is from importance to details that matter. That is, describing and evaluating the performance of the most important variants of the system that we have gradually built, first. Then moving to variants that bring further performance improvements, but that are less crucial. This way of description also means that we are revealing our work in a bottom-up way in terms of network layers. That is – from MAC (Medium Access Control) layer towards application layer. This organization also follows the natural (in our case) dependence of the application layer on information about the wireless link status that is provided by the MAC layer.

A significant part of the thesis is organized as chapters that are based on, or include published papers. These are Chapter 3 and Sections 4.2 and 4.3 of Chapter 4, and they can be read separately. The rest of the thesis is in a standard form.

Chapter 2 gives an overview of the QoS-related efforts that exist so far in the Internet and of Link Adaptation for wireless links in particular. A brief introduction to the IEEE 802.11 standard for wireless data communications, important for further understanding the material, is also presented.

In Chapter 3, the idea of our novel hybrid rate-controller is introduced. Then the practical implementation and the performance of the controller are discussed.

Chapter 4 focuses on cross-layer communication to support application-level adaptation. A model used to calculate the user-available throughput is introduced. Two different scenarios that are using this model are then discussed. The first scenario is one where the medium is not shared with other users – a typical situation for wireless home networks. The second is a scenario where we have multiple users sharing the same medium – a typical situation in a public access network.

Chapter 5 is a description of our RA*I*L API, which resulted from our experiments reported in Chapters 3 and 4.

Finally, Chapter 6 summarizes our work, offers final remarks, and discusses the possibilities for future work.

Chapter 2

QoS-related efforts in wireless links

Over the past decade and a half Quality of Service (QoS) was a research topic of many papers concerning networking. Most of the early studies were concentrated on QoS in then mostly wired Internet. Internet is based on the best-effort service model and this is insufficient for many types of real-time applications. So efforts have been made to create extensions and changes in Internet so that *certain types* of applications can be given end-to-end *assurances* about the network performance. This gives the definition of the term Internet QoS – *providing service differentiation and performance assurance for Internet applications* [Zha00]. Generally there are two kinds of service differentiation – per-flow or aggregate. A flow is a sequence of packets sent from a particular source to a particular destination for which the source desires special handling by the network. On the other side, in the aggregate approach packets are divided into several groups, called traffic classes, having different QoS levels. It is assumed that packets in the same class have similar QoS requirements, regardless the flows they belong to.

A number of approaches and mechanisms were designed to make Internet QoS-aware; the most notable per-flow example is Integrated Services with Resource Reservation Protocol (IntServ/RSVP). Example of an aggregate approach is Differentiated Services (DiffServ). Other efforts that do not fall strictly into above categories are Multiprotocol Label Switching (MPLS), traffic engineering and constraint-based routing [Xia99]. Why then, despite its 10 years history Internet QoS is still not in service [Sch01]? The reasons for that can be classified in two groups. First, **we do not really need it**. Over-provisioning (although not providing quality assurances) has proved a working method to deal with quality; fiber is cheap and there is an excess capacity in the backbone, so fixed Internet does not need QoS. Second, **the price/benefit ratio is too high**. The contradiction between complexity/overhead/reliability on the one hand and the strictness of the QoS guarantees on the other cause the price-to-value ratio of most schemes to go high, thus making them unattractive for deployment. Operators do not like any technology that is not scalable, not relatively easy to integrate, unreliable, that cannot be managed, audited or controlled and that cannot be charged or sold. To summarize, QoS in wired networks is unlikely to see any more progress.

In comparison to the (wired) Internet, wireless technology has an excess of problems that need to be resolved. Low bandwidths, high error-rates are common for the wireless environment. Users move and environmental parameters change, causing variations of the throughput and the latency. Also handovers (i.e. connection transfers to another channel or Access Point) occur and this causes connection drops. While download applications are usually fine with those effects, as far as the average throughput does not drop significantly, such issues are not tolerated by real-time traffic.

In the wireless domain over-provisioning is not a solution, both because of the necessity to share the medium with others and because of the limited energy sources that a mobile node has. As a result, quality is generally lower than in a wired environment, and varying over time and location. Therefore QoS mechanisms are mandatory in (mobile) wireless networks.

Looking at the network stack, there are different approaches where to apply QoS. It should definitely be done at the link layer - there it is actually a link adaptation. Without link adaptation there is no proper connection at all. At the application layer QoS should be applied as well – according to the end-to-end QoS argument. The principle of end-to-end QoS suggests that “functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level” [Sal84]. There is no use of doing something at layers in between, because of one of the same arguments against QoS in Internet: It should be supported at the other end as well - i.e. the compatibility price becomes too high. At the same time the benefits are too low, resulting in low benefit/price ratio.

Another possibility for QoS improvements is by making network layers exchange QoS-related information between themselves in an attempt to help each other handle better changes in link quality. We also have established that cross-layer interactions are quite beneficial, but only in the case they are well engineered. That is – significant profit/price ratio can be achieved only if the factors influencing the link quality are sufficiently well understood, accounted for, and properly modelled (Chapter 4).

In the world of wireless data networks IEEE 802.11 has a dominant role, and it has still the QoS issues discussed above unresolved. Therefore we picked up 802.11 as a target technology of our research. The next section gives some background information about 802.11. Further on in this chapter different QoS approaches are discussed, according to their position in the network stack. Finally, concluding remarks are given.

2.1 Introduction to 802.11

If the reader is not familiar with 802.11, it is recommended that she or he reads this section, since all the research described in this thesis is closely tied to the 802.11 wireless networking technology.

2.1.1 History and overview

Research on wireless networks (based both on infrared and radio) existed since late 1970s [Pah95], but for a long time it did not result in any commonly-accepted, publicly-available technology. The main reason for that, at least concerning the research of wireless networks relying on radio, was the lack of commercially available frequency bands. After the ISM (Industrial, Scientific and Medical) bands were allocated for more flexible (that is - unlicensed) use in 1985, companies got interested and more serious research concerning Wireless LANs followed. The fact that ISM bands were unlicensed, that is - no registration and payment are necessary, meant that a set of rules had to be imposed to avoid abuse. These rules stated that first, limited power should be used for all transmissions, and second, a Spread Spectrum [Mil97] type of modulation should be used. Spread spectrum is a radio frequency modulation technique where the radio energy is spread over a much wider bandwidth than really needed for the data rate. This is done to increase the immunity of a system to interferences and consequently to ease the device/technology coexistence. These modulation type constraints predetermined the core radio characteristics of the 802.11 standard. After the IEEE 802.11 group was formed in 1990, it took it more than 7 years to finalize the standard. The main reason was that every vendor participating in the group was trying to push the standard towards its own technologies. One of the successful early-1990s commercial wireless network devices, however, managed to produce significant impact on the 802.11 design. The device was WaveLAN®, and was introduced in 1991 by NCR. Although initially working in the 915 MHz band and having a number of shortcomings (i.e. bulky, high-priced and power hungry), it underwent a number of upgrades and improvements, including the move to the 2.4 GHz band. Eventually, at the time of publishing of the 802.11 standard, so many WaveLAN features would be adopted by the standard, that WaveLAN would turn out to be almost a 802.11 device. Similar was the case with WaveLAN-II, developed by Lucent Technologies [Kam97]. It was released just before the official publication of 802.11b, and although technically not an 802.11b device, it incorporated all the 802.11b features, plus some more. This caused a widespread confusion and mixing of the terms “WaveLAN” and “802.11”, which persists even today.

The initial frequency band 802.11 was designed to work in is the 2.4 GHz ISM band. The PHY (physical layer) specifications in 802.11 also included an IR (infrared) variant for a medium, but it did not make it to the market, so we will not discuss it here. Following the FCC rules for ISM bands, 802.11 uses two spreading techniques to allow for peaceful co-existence of different technologies. These spreading techniques are [Int00a; Agi01]:

- **FHSS** (Frequency Hopping Spread Spectrum) uses a number of narrow channels (79 for FCC and ETSI regulatory domains) that the system switches between in a relatively short time (fractions of a second), and in a pseudo-random fashion. There are a number of different hop sequences, so several Basic Service Sets (BSS)[1] can coexist with a relatively low chance of collisions on some channel.

- **DSSS** (Direct Sequence Spread Spectrum) uses a single wide channel, in which the spreading is achieved by XOR-ing the data with a higher-speed pseudo-random numerical sequence (called PN).

By the time the original 802.11 standard was published, it was already clear that it was substantially lagging behind wired networks in terms of throughput (less than 2 Mbits/s versus 100 Mbits/s). Therefore, in 1999 two of the most important addendums – 802.11a and 802.11b – were produced. Both aimed at increasing the data rates, they achieved the goal in a very different way.

802.11b built on the DSSS variant of 802.11 by playing smart tricks like CCK (Complementary Code Keying) [Int00b], and thus adding two more rates to 802.11's 1 and 2 Mbit/s – 5.5 and 11 Mbit/s (see Table 2.2). By betting on DSSS, FHSS was completely abandoned [Spa00].

Only 3 non-overlapping channels out of total 13 available (11 in USA) exist for 802.11b (see Table 2.1). This makes it difficult for different BSS to coexist, relying mainly on attenuation as separating factor, rather than on availability of channels.

Regulatory domain	Channel Center Frequency, MHz	Channels	Maximum EIRP[2], mW (dBm)
ETSI (Europe)	2412 – 2472	1 – 13	100 (20)
FCC (USA)	2412 – 2462	1 – 11	4000 (36)
Japan	2412 – 2472	1 – 13	10 per MHz (0)

Table 2.1: *Frequency allocation for 802.11b/g.*

[1]A Basic Service Set is the cell that is served by a single Access Point.

[2]**Effective Isotropic Radiated Power**. To provide a common reference for emitted power, an ideal isotropic antenna is used as a base. An isotropic antenna is a singular point (dimensionless), whose wavefront is a perfect sphere of constant voltage (or power for equal impedances). Any gain specified for a real antenna represents a concentration of the radiation pattern in a given direction. EIRP is calculated as $EIRP = P_o - L_f + G_a$, where P_o is the power at the output of the transmitter, L_f denotes the losses in the feed line, and G_a is the gain of the antenna (all the terms are expressed in dB-related metrics).

Radio Data Rate, Mbits/s	Modulation scheme
1	DBPSK
2	DQPSK
5.5	CCK+DQPSK PBCC+DBPSK (optional)
11	CCK+DQPSK PBCC+DQPSK (optional)

Table 2.2: *Radio data rates and modulation for 802.11b.*

802.11a took an entirely different approach from 802.11b. It not only moved to a new frequency band, the 5 GHz U-NII (Unlicensed National Information Infrastructure) band (see Table 2.3), but it also employed a new modulation technique called OFDM (Orthogonal Frequency Division Multiplexing). There are three advantages of the 5 GHz U-NII band compared to the 2.4 GHz ISM band. First, it is much less EMI (Electromagnetic Interference) polluted – there are no microwave ovens emissions or alike there. Second, there are many more channels available (in Europe 19 versus 3 non-overlapping for 802.11b). Third, U-NII bands do not have the requirement that a spread spectrum technology must be used. This is where OFDM comes into action. This technique uses multiple carriers at orthogonal frequencies, to transmit data simultaneously. In the case of 802.11a there are 52 carriers, of which 48 are used for data, and 4 are pilot carriers. In this way, the monstrous (for 1999) 54 Mbit/s maximum data rate is achieved (see Table 2.4).

Regulatory domain	Channel Center Frequency, MHz	Channels[3]	Maximum EIRP, mW (dBm)
ETSI	5180 – 5320 5500 – 5700	36 – 64 100 – 140	200 (23) 1000 (30)
FCC	5180 – 5240 5260 – 5320 5500 – 5700 5745 – 5805	36 – 48 52 – 64 100 – 140 149 – 161	200 (23) 200 (23) 1000 (30) 4000 (36)
Japan[4]	5180 – 5320	36 – 64	100 (20)

Table 2.3: *Frequency allocation for 802.11a*

802.11a has three big disadvantages though, two of them non-technical, which caused its still continuing low popularity. The first is the sluggish appearance

[3]Each fourth channel is used, like 36, 40, 44, etc.

[4]Regulations for Japan are likely to be changed, since the Japanese regulatory agency MPHPT is supposed to enable the 5470 – 5725 MHz bands progressively in near future.

Radio Data Rate, Mbits/s	Modulation scheme and coding rate	
6	BPSK	1/2
9	BPSK	3/4
12	QPSK	1/2
18	QPSK	3/4
24	16-QAM	1/2
36	16-QAM	3/4
48	64-QAM	2/3
54	64-QAM	3/4

Table 2.4: *Radio data rates and modulation for 802.11a*

of 802.11a products on the market. The reasons were the technical difficulties in implementing OFDM, and the reluctance of the companies to move on with the new technology because of the second disadvantage. The latter is the total incompatibility of 802.11a with 802.11 and 802.11b products. Humans show remarkable conservatism for totally replacing a proven working device (geeks make exceptions here), especially when they are expected to make serious investments in something untested. Users like to move on gradually, thus putting strong preference on backwards compatible technologies like 802.11b. The third disadvantage of 802.11a is its shorter range compared to 802.11b, especially for indoor environments. The reasons are the greater spatial attenuation at 5 GHz, especially indoors, and the higher bitrates, which require higher SNR (signal-to-noise ratio) thresholds at the receiver.

Another, smaller disadvantage of 802.11a is its bigger hunger for energy, since devices using OFDM are generally consuming more power than devices that use Spread Spectrum.

The ever increasing demand for more throughput contradicted the compatibility-driven choice to stick to 802.11b more and more vividly, causing a deep deadlock. Something had to be done about it, and the solution came in 2003 in the form of 802.11g. In a nutshell, 802.11g is a 802.11a shifted to operate at 2.4GHz. It uses the same modulation scheme (OFDM) to achieve the same high rates (up to 54 Mbit/s) as 802.11a. It has also some add-ons to be backwards compatible with 802.11b. The result is an even messier 2.4 GHz band, and questionable performance results. Some analysts claim that 802.11g might kill 802.11a, because it "gives the same performance", but we think that actually 802.11g will be the bridge to using 802.11a. The reason is that now most of the chipset manufacturers have experience with OFDM, because they had to produce the 802.11g devices, whether reluctant or not. From there it takes only one minor step to 802.11a, which is proven by the rapidly increasing release of multimode a/b/g chipsets. With these already in their mobile devices, people will start migrating to 5 GHz where there is still a lot of fresh air.

Since the original publication of the core standard in 1997 and the important 802.11a and 802.11b annexes in 1999, the IEEE 802.11 working group franticly produced a big number of additional Amendments [IEEwg], also often referred to as the 802.11 Alphabet Soup. Some of the amendments that are considered important by the author, apart from the already discussed 802.11a/b/g, are briefly summarized below:

802.11e Brings QoS enhancements to 802.11. This is done by introducing traffic classes, which can be used for prioritizing specific traffic, such as streaming. In a more advanced configuration, 802.11e also allows for precise control of bandwidth, fairness, and packet jitter. Although the addendum shares one of the goals of this thesis – to improve QoS for streaming applications – the way 802.11e achieves this objective is different, and still dependant on how well LA performs. So, this amendment can be a complement to our work, but it cannot replace it.

802.11h Introduces the possibility of transmission power control in the 5GHz band, and can be considered as add-on to 802.11a. It may be important for Europe because of its radio regulations.

802.11i This amendment provides improved security over the defeated WEP (Wired Equivalent Privacy) specification. The improved WPA (Wi-Fi Protected Access) security mechanism, which was introduced by the Wi-Fi alliance and used some of the 802.11i available components at that time, is considered an interim standard.

802.11k This amendment is related to our work. Originally 802.11 did not specify if and how radio status is propagated to higher layers. This extension will provide interfaces for providing radio resource measurements like Received Signal Strength Indication (RSSI).

802.11n An addendum that aims at increasing the throughput available to users beyond 100 Mbps, most likely by using MIMO (multiple input/multiple output) techniques.

802.11p An extension to support communication in vehicular environment. Speeds of up to 200 km/h are targeted.

802.11r The amendment will provide for faster handoff between Access Points, thus improving the roaming performance.

2.1.2 Basic MAC concepts

The MAC, or the Medium Access Control, is responsible for governing the usage of the scarce resources of the radio channel. Essentially, the MAC determines the way each node can access the channel to transmit information. Before addressing the 802.11 MAC, the most popular access methods employed in wireless networks will be discussed briefly.

FDMA (Frequency Division Multiple Access) separates users by assigning them to different portions of the radio spectrum. This is one of the oldest methods, and is actually employed as a primary access method by most of the wireless technologies. In practice, FDMA takes form of different channels that wireless devices can choose from (for example, three non-overlapping channels in the case of 802.11b).

TDMA, or Time Division Multiple Access, allows each user to occupy the channel for a specific period of time. Usually systems with TDMA employ a central station to coordinate the other nodes. The time is divided into slots, usually organized in groups called frames, and each node is assigned certain number of slots for the transmission of its data.

CDMA (Code Division Multiple Access) is a scheme that is applicable only for systems using DSSS as a modulation technique. Unlike FDMA and TDMA, CDMA does not separate users in the frequency or time domain – everyone uses the full available bandwidth all the time. Instead, CDMA achieves the goal of distributing the radio resource between users by assigning them different codes (i.e. the pseudo-random numerical sequence that spreads the signal). Multiple users (each using a different code) can access the medium simultaneously, since transmissions with the "wrong" code do not influence the reception of "proper"-coded transmissions.

Finally, CSMA (Carrier Sense Multiple Access) is the mechanism mostly used in today's wireless LANs. In 802.11 CSMA/CA (CSMA with Collision Avoidance) is used. The basic idea of CSMA is that each node listens before it attempts to transmit a message. If the medium is clear, it goes on and initiates a transmission. In the case the medium is busy, it waits until it clears and then initiates a "contention" phase – waiting for a random period of time. If after that period the channel is still idle the station sends the message. Otherwise it will pick another random wait interval within a window (called contention window) that will increase with each failed attempt (behaviour known as exponential backoff). Usually the contention window is not extended anymore after reaching some threshold value, to improve the MAC stability in stress conditions, such as a very bad channel or an excess of users.

There is a difference between 802.11 and Ethernet, the latter using CSMA/CD (CSMA with Collision Detection). A CSMA/CD system can detect a collision any time it occurs, while a system employing CSMA/CA can determine if there was a collision not until after a packet transmission is over. Therefore CSMA/CD is more efficient than CSMA/CA. However, CD is difficult and impractical to implement with radio technologies. The reason is that most radios, including 802.11, are half-duplex, that is, they cannot listen while transmitting [Int99]. Therefore they cannot detect a collision, i.e. another station transmitting at the same time. But even if a radio listens while transmitting, its own signal would normally mask the signals from other radios, since these signals attenuate while travelling to the station in question. Cancellation of a station's own signal in the receiver is possible, however complex and therefore expensive, so normally it is not done.

2.1.3 Advanced MAC

In 802.11 packets are encapsulated in units called MSDU (MAC service data unit), which are then transmitted over the air. To avoid confusion, we will further refer to data units on the link layer (MSDU's) as frames, and on the layers above – as packets.

The chance of collisions and the overall unreliability of the radio channel make the probability of successfully sending a message over the air relatively low. So, if there is not an additional mechanism to take care of retransmitting previously failed frames, the packet loss would be unacceptably high. Protocols like TCP are not a solution in this case, because they assume packet losses are due to congestion, and their efforts to resolve the situation would reduce the performance. Therefore, the retransmission mechanism is implemented in the MAC layer. Positive acknowledgements are used to assist retransmissions. When a station receives a frame, it sends back a small message (ACK) to indicate that the reception was correct. A typical IEEE 802.11 send/acknowledge procedure is shown in Figure 2.1.

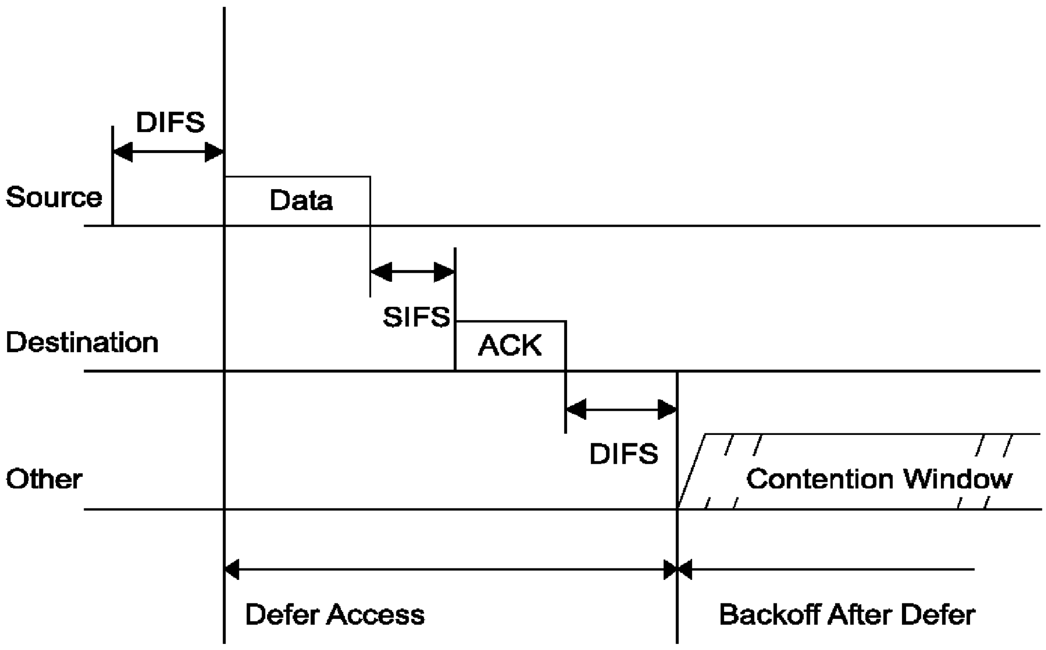


Figure 2.1: *Basic access mechanism*

In 802.11 the time interval between frames is called IFS (Inter-Frame Space). There are four different types of IFS, which are used to prioritize transmission of certain frames and in this way specific frame sequences are forced to occur. In this case SIFS (Short IFS) is – as it name implies – shorter than DCF IFS (DIFS). DCF is the acronym for Distributed Coordination Function and is explained later. By using SIFS between a frame and its acknowledgement, it is assured that the ACK will be transmitted before any attempt by another station to access the medium can be made.

If after sending a frame a station does not receive an ACK, it assumes that the frame was lost (or not received correctly by the destination node) and retries to send the same frame again. This continues until a certain count (retransmit limit) is reached, after which the node gives up sending this frame and continues with transmission of the next.

The probability of a frame being sent correctly is inversely proportional to its length. Therefore 802.11 employs a feature called fragmentation that allows a frame to be divided into smaller pieces (MPDUs - MAC protocol data units), which are then transmitted separately. There is a packet size threshold that is determined by the user, which indicates what is the minimum size above which a packet should be fragmented. While fragmentation indeed improves the transmission reliability of large frames in situations with interference or weak signals, it also reduces the maximum throughput available at the user-level due to increased MAC overhead. Therefore packet fragmentation should be only used in noisy environments. Unfortunately, the decision to use or not fragmentation, in most products is still left to users discretion, which usually means that it is not used at all (people do not care about it). Thus, the problem of automatic fragmentation threshold selection is still open to research community.

There is a specific problem that exists with wireless networks - the hidden terminal problem. Consider for example three stations, A, B, and C, as shown in Figure 2.2. B is in range of both A and C, but A and C cannot hear each other: because they are too far away or maybe because there is some obstacle that is blocking the communication between them. If A transmits a message to B, C will not be aware of that and could transmit as well. The result will be a collision, so B will get no message.

A solution to this situation exists in 802.11 – this mechanism is called RTS/CTS (Request to Send/Clear to Send). In essence this is a handshaking mechanism. A station would send an RTS request before it tries to send a packet (see Figure 2.3). If the intended receiver senses that the medium is idle – that is, not used by other stations except the one sending the RTS – it will send a CTS back. Thus, in our hidden-terminal situation, C will hear the CTS by B, and will not attempt transmission – the collision between A and C will be avoided. RTS/CTS is also used to implement *virtual carrier-sense*. The latter is a reservation mechanism for announcing forthcoming usage of the medium. The entity that holds that reservation information in 802.11 is called the network allocation vector (NAV).

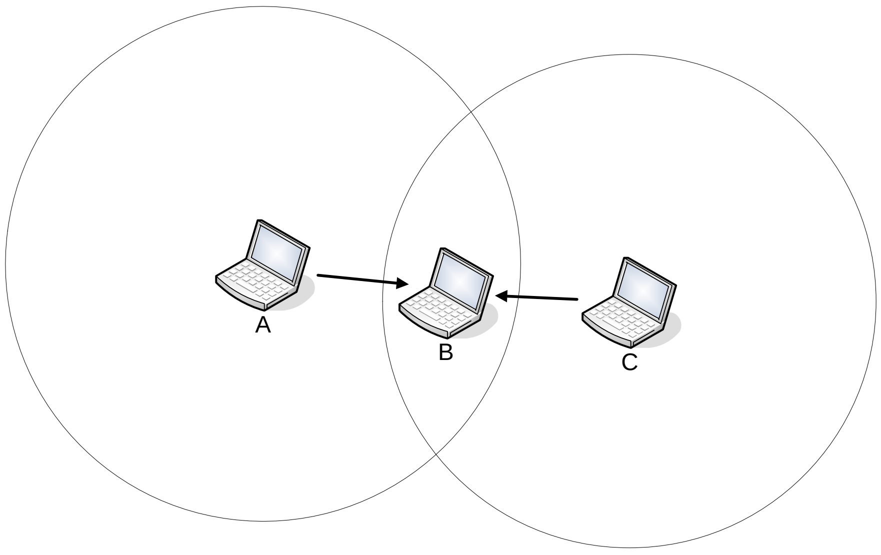


Figure 2.2: *Hidden terminal problem*

The NAV contains information about future traffic. Each station updates its NAV based on duration information that is present in RTS/CTS frames prior to the actual data transmission.

RTS/CTS creates another problem though, called the exposed terminal problem (Figure 2.4). Here only B is in range of A, C and A are in range of B, and only C is in range of D. When A transmits to B, the latter replies with CTS, which is heard by C. Then, if D wants to transmit to C, which is permissable since both communication can go without disturbing each other, it will not be allowed because C will hear the CTS by B, and will not reply to D with an own CTS message. This problem does not have serious performance implications, therefore it is not addressed in 802.11.

Another way of defining the medium access behaviour of the stations belonging to a BSS, is by introducing a logical function called "coordination function", as defined by the 802.11 standard [IEE11]. There are two such functions defined in the standard – the DCF (distributed coordination function) and PCF (point coordination function). DCF is the fundamental access technique, and is the already mentioned CSMA/CA with ACKs method. PCF depends on DCF and is used in infrastructure mode only, because the need of a **point coordinator** operating at the access point. This coordinator has the task of deciding which node has the right to transmit at any given moment.

Essentially the combination PCF plus DCF (PCF cannot exist alone, because relies of the services provided by DCF) is a combination between TDMA and CSMA.

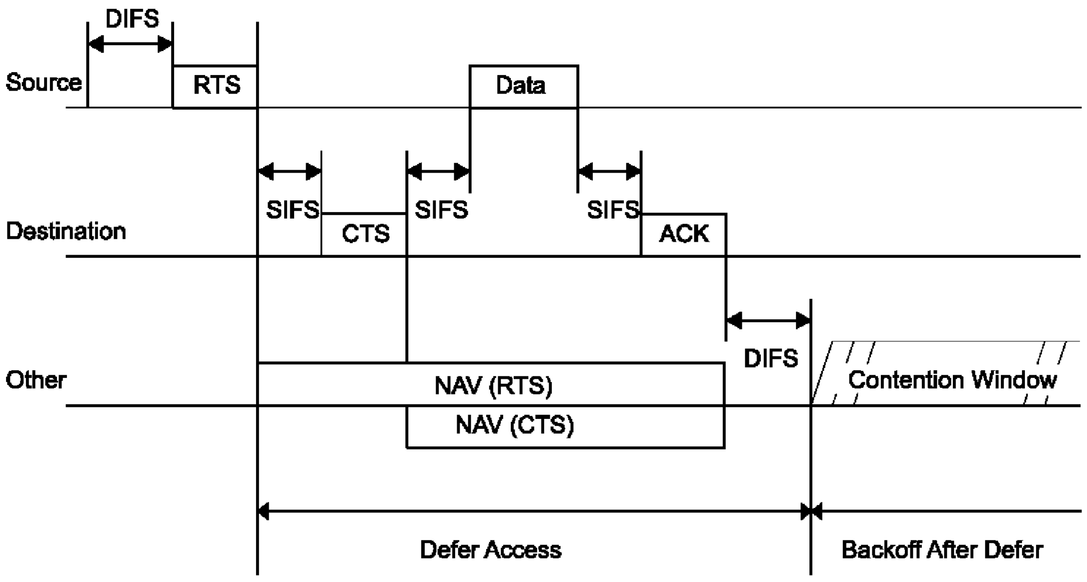


Figure 2.3: *RTS-CTS mechanism*

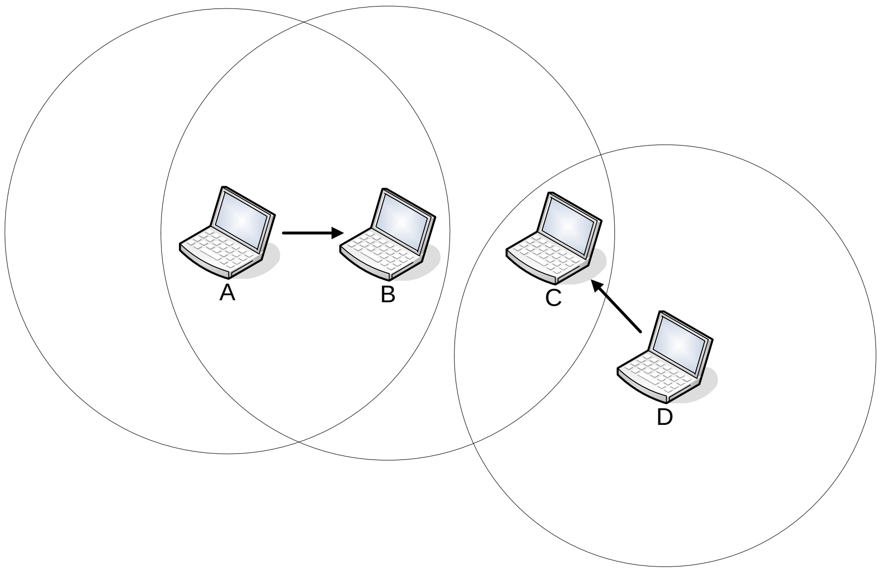


Figure 2.4: *Exposed terminal problem*

2.2 Types of Adaptation according to network stack position

There are different approaches to mitigate the inherent problems of radio links – the ones that were discussed in the beginning of this chapter. The greatest fundamental differences are between methods that operate in different network layers. This is due to the very different control context in each layer – that is the different feedback information available, the different control variables and the different ways to modify them. Therefore the presentation of the different adaptation methods in this section is made according to their position in the network protocol stack.

2.2.1 Link Adaptation in 802.11

At the bottom of the stack – on the wireless link layer (PHY) – the important process of Link Adaptation (LA) takes place. LA is the act of changing the parameters of the wireless link, following the changes of the link conditions. The task is to achieve the optimum performance of the link in terms of throughput and packet delay, at any given moment. This implies amongst others that LA should take care that a station stays connected as much as possible.

LA is a scheme that is typically employed in wireless systems that involve mobility, since the latter has the greatest impact on link conditions and, hence, performance. Therefore you can find LA in most mobile phone networks (like GSM, UMTS, etc), and in wireless data networks like 802.11 and HIPERLAN [Lin00]. In GSM for example, Tx RF Power and Timing Advance are adjusted according to the link conditions and distance from a mobile to the Base Station [Han97]. The Timing Advance is the transmission delay used to compensate for the time it takes the signal to travel between the Base Station and the mobile. This delay is very important in systems employing TDMA on large dynamic ranges of distance, such as the 35 km cell radius of GSM for example.

In 802.11 LA was left completely out of the scope of the standard. It was left to vendors to implement it as they like. This decision may seem strange for a system that exercises highly mobile radios. We think this decision is due to the very limited choice of values at that time for the most important parameter in 802.11 – that is the rate setting. In the original standard only 2 data rates exist – 1 and 2 Mbits/s. The later addendums 802.11b and (especially) 802.11a introduced more rates (4 for 802.11b and 8 for 802.11a), but surprisingly, the LA issue was still not addressed.

There are several link/MAC-level parameters that can be controlled in 802.11. Examples are the maximum number of retransmissions, RTS/CTS scheme enabled or not, and packet fragmentation threshold. It was already mentioned that the most important parameter available for adjustment is the transmit rate. Each rate corresponds to a different modulation scheme and provides a way to balance throughput and connectivity at various degrees of link deterioration. The latter could result for example from increased distance between stations, or from interference by another radio device. The larger throughput a modulation scheme provides, the better conditions (greater signal strength at the receiving station) it needs to achieve a certain bit error rate. The task of LA mechanisms here is, by switching between rate settings, to keep the connection up under various operating conditions while maximizing the throughput.

All closed-loop control algorithms need information about the state of the system the algorithm is in control of – in other words: the controller needs feedback. Regarding wireless systems, this feedback is the information about the link conditions, or the channel state information (CSI).

The most important CSI indicator is the SNR (signal-to-noise ratio), since it directly determines the maximum theoretically available channel capacity, according to Shannon's theorem [Sha48; Int97]:

$$C = B * \log_2(1 + S/N) \tag{2.1}$$

where C is channel capacity in bits/s, B is the channel bandwidth in Hz, and S and N are the signal and noise strength respectively, in watts. In reality, how closely a wireless system can approach this capacity limit depends on the modulation technique. Therefore the practical aspect of the SNR feedback is that based on the SNR value the current level of reliability in terms of BER (bit error rate) for each modulation can be determined, and consecutively, the appropriate rate can be selected.

From the control theory point of view, when an LA algorithm gets a CSI directly, then it is actually a *feedforward* control system. This is because for LA channel state is a disturbance, which can only be measured, but not controlled (see [Lev96], page 208). The advantage of the feedforward approach is that a controller "can use a disturbing variable (CSI in our case) to manipulate a correcting variable (rate setting) directly, without waiting for its effect on the controlled variable (FER, throughput)" ([Lev96], page 1216). This results in more responsive Link Adaptation for the algorithms that use SNR, or SNR-related information (like SSI - signal strength indication), as a CSI indicator. But since the target audience for this work is the computer science community, to avoid confusion, all the information that comes to the LA controller from the physical layer, will be further addressed as feedback.

SNR is hard to measure directly in 802.11 radios, because it requires calibrated circuitry that would be too expensive to build in such cheap devices. Therefore most of the rate-control methods use indirect methods to get CSI, that is – methods that rely on measuring the current reliability of the channel by obtaining some statistics related to BER, FER (frame error rate) or PER (packet error rate). Thus, from the control theory point of view, in this case the control becomes a true feedback control, because the link reliability depends on the controlled variable (the rate setting). Since the way CSI is obtained (i.e. resulting in feedforward or feedback scheme) determines key characteristics of the rate-controller such as responsiveness, accuracy and stability, which then reflect to the quality performance of the system, we classify the rate-control algorithms according to the type of feedback.

The majority of rate-control algorithms use *statistics-based* feedback. To the author's knowledge virtually all 802.11 products employ this flavor of LA. There are three basic types of statistics-based rate control: throughput-based, FER-based, and retry-based. The first approach uses the most global type of statistic (upper layer perceived throughput) and has the slowest response. Slow response causes communication drop-outs when the link conditions degrade rapidly (e.g., when the user moves fast), and these drop-outs are not handled well by streaming applications. The retry-based control uses the most local statistic (number of retries per frame), and is the fastest method [Kam97; Veg02; Lac04]. However, since it cannot be determined if the cause of a retry is low SNR, or a collision, this type of control has disastrous effect in loaded environments on both current and other users' throughput [Heu03]. In the FER-based approach, the Frame Error Rate (FER) of the data stream transmitted over the link is used to select an appropriate rate. This method is somewhere between the previous two approaches with advantages and disadvantages coming from both throughput-based and retry-based rate control algorithms.

The other basic class of rate-control algorithms uses direct, *SNR-related* feedback. Usually it is uncalibrated signal strength indication (SSI). This approach has been studied in previous work [Bal99; Ue98; Hol01; PP03], but to the author's knowledge, until the moment of writing of this work, there is no practical implementation in any commercial 802.11 product. Although the lack of implementation does not suffice for carrying out new research, it indicates that something is wrong. There is a problem that needs to be addressed before a rate-control algorithm using SNR-related feedback can be employed in a 802.11 device. IEEE 802.11 was conceived as an affordable way to get mobile Internet connectivity, without having to rely on expensive infrastructure. This means that both ends of a 802.11 link, being two stations (ad-hoc mode), or a station and an access point (managed mode), are made of relatively cheap hardware. The result is that the quality of a SSI feedback that you can get from those devices is low, meaning biased, noisy and drifting readings. This would lead to poor rate-control, if the methods from the previous work mentioned above are directly implemented. Of course, it can be argued that methods such as Kalman filtering can be employed for processing the SSI feedback, as it is done in other communication systems like GSM, GPRS and UMTS [Leu01], but unfortunately this cannot be the case with 802.11. The reason is that while in GSM for example, such processing is done at the Base Station, which is an expensive equipment with enough spare processing power, in 802.11 this processing should go to the 802.11 firmware or device driver. It is clear that in the latter case, computationally intensive methods are unapplicable, so another solution is needed. One of the major contributions of this work is the practical implementation and testing of an advanced hybrid rate-selection algorithm, using both SSI and statistics-based feedbacks (see Chapter 3).

2.2.2 Other QoS efforts on the Link layer and on the Transport layer

Significant efforts are made by the research community to enhance existing QoS schemes or to introduce new ones on the link and network layer, that will cope with the new challenges introduced by the wireless link.

Efforts at the link layer included modifications to the MAC so that multiple channels are used for retransmissions (the SMPT approach [Fit98]). Also recently differentiation mechanisms were added to 802.11 MAC, in a standardization effort [IEE11e] to address QoS there.

Concerning the transport layer, it was quickly discovered that while TCP is doing quite well in wired Internet, it produces disastrous results when employed in wireless networks [Cha96; DeS93; Pil03]. The reason is that because wireless channels suffer from bursty error losses, they trick TCP to incorrectly assume network congestion and back off, resulting in drastic reduction in throughput. Therefore most of the research in this area concentrated on modifying TCP [Bal02; Gun02; Jia01] or on hiding losses from it [Aya95; Bak95; Bal95; Sin02].

2.2.3 Application-layer based adaptation

Following the end-to-end QoS argument [Sal84], which was already discussed earlier, it is important that (multimedia) applications take measures to adapt to changing network quality.

The normal QoS scheme that works with wired links – buffering – produces unsatisfactory results when applied to a highly variable wireless link. Therefore additional methods are studied by researchers. The most common approach is to vary compression parameters (such as the quantization level) so that the data rate produced by the video/audio encoder follows the available network bandwidth. Another approach when using layered video is to drop layers accordingly. If there is information available about the error rate, the application can change the error protection used.

An important issue with application-level QoS is the way the network state feedback is obtained and the quality of the feedback. Some of the QoS schemes use indirect methods like network performance observed at higher network layers (above data link layer) to get an idea about the state of the wireless link [Nun04]. While simple, this method yields the worst results since the information is usually too late and too inaccurate. Other approaches use direct feedback such as SNR, which gives good results, but the price to be paid is bad compatibility across different network cards [Hu02].

2.3 Cross-layer interaction

Application-layer approaches need some kind of feedback about the link quality, so there is always some kind of cross-layer interaction.

There are two major approaches in cross-layer interaction – informational only, and bi-directional.

In the first class (informational only), which is the most widely used, upper network layers utilize feedback information generated by the link layer. Such information includes details about the current channel conditions like PER (packet error rate), resulting available throughput, average packet delay and so on.

In the second class (bi-directional), some authors go further and exploit top-down information exchange as well. These can be simply instructing the link layer what to do. Or, it can also include negotiating parameters between layers. An example is the ARC approach [Dij00], where each network layer negotiates abstract QoS parameters with the lower and the upper layer, until a global optimum is obtained.

2.4 Concluding discussion

QoS efforts exist at all network layers, but it is the link and the application layers where these efforts make most sense and produce the most significant results. This is easy to explain, if having in mind a rule-of-thumb in control that says that a controller should be placed as close to the problem as possible, and where it can obtain the best feedback. Therefore, the presence and behaviour of Link Adaptation is critical in wireless systems because it handles the bottleneck - the unreliable, ever changing radio channel. Less critical, but still important is the application-layer QoS adaptation, since there lies the entity (the video encoder) that knows best what the QoS requirements are regarding the performance of the whole system, as perceived by the user – the ultimate judge.

Chapter 3

MAC rate control

Streaming multimedia content in real-time over a wireless link is a challenging task because of the rapid fluctuations in link conditions that can occur due to movement, interference, and so on. The popular IEEE 802.11 standard includes low-level tuning parameters like the transmission rate. Standard device drivers for today's wireless products are based on gathering statistics, and consequently, adapt rather slowly to changes in conditions. To meet the strict latency requirements of streaming applications, we designed and implemented an advanced hybrid control algorithm that uses signal-strength (SNR) information to achieve fast responses. Since SNR readings are quite noisy we do not use that information to directly control the rate setting, but rather as a safeguard limiting the range of feasible settings to choose from. We report on real-time experiments involving two laptops equipped with IEEE 802.11a wireless interface cards. The results show that using SNR information greatly enhances responsiveness in comparison to statistics-based rate controllers. Finally, we will present the results of an experiment with realtime video streaming to a moving laptop in an office-like environment. Our hybrid control algorithm effectively prevented many packets losses, thereby achieving a much higher video quality than the statistics based algorithm.

Keywords: rate control, MAC layer, SNR, link adaptation, video streaming

This chapter has been published as "Automatic IEEE 802.11 Rate Control for Streaming Applications" by I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips in the special issue of *Wireless Communications and Mobile Computing (Wiley Interscience)*, on "Radio Link and Transport Protocol Engineering for Future-Generation Wireless Mobile Data Networks", Vol. 5(4), June 2005.

3.1 Introduction

It is anticipated that multimedia streaming over the Internet will have a significant share in tomorrow's communications. Also, end users increasingly seek mobility, thus paving the way for extensive deployment of wireless technologies like IEEE 802.11. The joint effect is that support is needed for multimedia streaming over connections that include both fixed and wireless links. In this chapter, we focus on the weakest part of such connections: streaming over a wireless link (the last hop). Such a link is the bottleneck for two reasons: First, communication over a wireless channel is simply not able to achieve the same quality (throughput, error rate, etc.) as its wired counterpart, which reduces the quality of the multimedia content that can be delivered. Second, in a mobile environment, the channel conditions can change rapidly due to changing distance between the stations (user mobility), Rayleigh fading, interference and so on. Since multimedia streaming applications must deliver their content in real time, they are very sensitive to jitter in packet delivery caused by retransmissions in the underlying transport protocols. Consequently, when using streaming applications, users experience reduced range compared to the case when less demanding applications like file downloading and web browsing are used.

With today's 802.11 products, the fundamental problems of wireless communication are aggravated by poor handling of the limited and imperfect resources (scarce spectrum, noisy medium) available to the radio. In particular, current transport protocols and device drivers do not actively control the user-available parameters of the 802.11 MAC layer; they use some default values instead. In this chapter, we demonstrate that much can be gained by tuning the MAC parameters to the (fluctuating) channel conditions.

The remainder of the chapter is organized as follows. The basics of the link adaptation are discussed in the next section. Section 3.3 gives a description of the existing rate control algorithms. Our improved solution is introduced in Section 3.4. The experimental results are discussed in Sections 3.5 and 3.6. The conclusions and future plans are presented in Section 3.7.

3.2 Link adaptation basics

The IEEE 802.11 standard defines several MAC-level parameters (settings) that are available for tuning at the side of the Wireless Network Interface Card (NIC). The most important parameter available for adjustment is the transmit rate. In 802.11a, for example, it can be set to 6, 9, 12, 18, 24, 36, 48 and 54 Mb/s. Each rate corresponds to a different modulation scheme with its own trade-off between data throughput and distance between the stations. This can be seen in Figure 3.1 - for clarity only the last four modulation schemes are shown. It shows the performance in terms of the throughput for each modulation scheme available

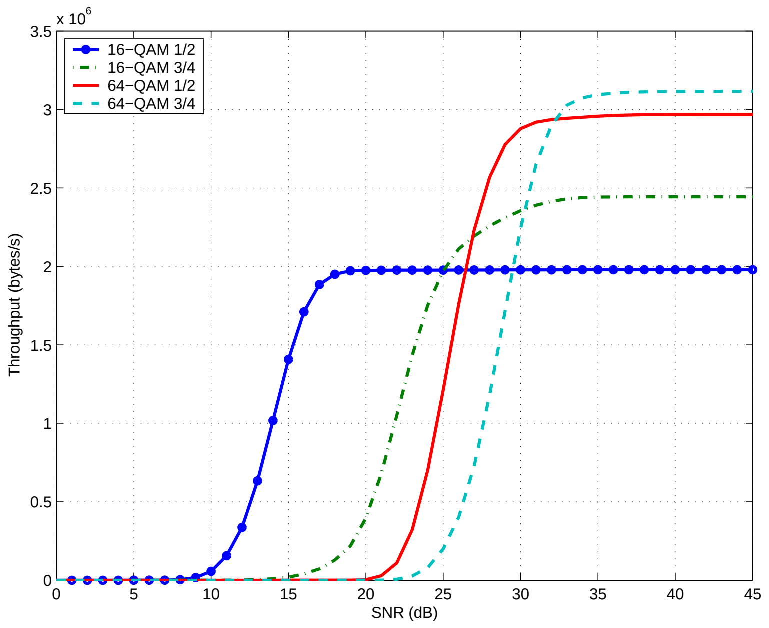


Figure 3.1: *Throughput vs. SNR for some 802.11a modulation schemes.*

in IEEE 802.11a versus the signal-to-noise ratio (SNR). Note that distance is related to SNR as $SNR \sim \frac{1}{dist^{\alpha}}$. More complex modulation schemes like 64-QAM 3/4 offer a larger throughput, but also have increased sensitivity to channel noise, and thus provide a shorter operating range. Usually, one wants to extend the operating range as much as possible and, at the same time, to maximize the throughput. This can be done by proper (automatic) selection of the rate (modulation scheme) that gives the maximum throughput for certain conditions, for example, by selecting 64-QAM 1/2 at an SNR of 30 dB (Figure 3.1).

In order to decide which rate is optimal at each specific moment, a control algorithm needs information about the current link conditions, or so-called channel state information (CSI). Since it is difficult to get CSI directly, most rate-control algorithms use some form of statistics-based feedback, for example, user-level throughput (see Section 3.3). The main disadvantage of this indirect feedback is that it is inherently slow, causing communication drop-outs when the link conditions degrade rapidly (e.g., when the user moves fast). The short-term drop-outs are normally handled by frame retransmissions. This is acceptable for download applications whose main requirement is gross data throughput. The retransmissions, however, lead to a significant increase in (average) packet delay, and the variations in the number of retransmissions cause an increase in the jitter of the packet delay. Streaming applications are very sensitive to long packet delays and high jitter, and less sensitive to the overall throughput of the link (provided of course that this throughput is larger than the minimum throughput that the application requires). Consequently, streaming applications perform poorly under standard automatic rate control. In [Har03], for example, it is reported that switching from automatic rate control to a manually selected fixed rate extends the maximum distance between stations up to 40% for the flawless display of a video stream over an 802.11a wireless connection.

A logical way to cope with the slow accommodation characteristics of statistics-based feedback methods is to look for methods that use faster feedback, i.e., feedback that quickly provides up-to-date information about the channel status. Such a feedback – the SNR – has been theoretically discussed in previous works, but it has never been used in actual implementations. The main reason for this is that, in practice, it turns out to be very difficult to obtain a reliable estimate of the SNR (see Section 3.3). In this chapter, we discuss an advanced hybrid approach to mitigate the SNR-related problems, and report on a practical implementation of this approach in a novel automatic rate-selection algorithm for IEEE 802.11a wireless connections.

It is important to note that all the CSI-based rate control schemes have one very important disadvantage in common. In case of low or no traffic, the CSI known to the system becomes outdated, therefore disrupting the work of the rate control algorithm. There are two ways to cope with this problem. The first is to use safety mechanisms, like timers to invalidate the CSI after not being updated for certain period of time. When the CSI known to the algorithm has been marked invalid, the rate controller can either revert to some safe rate setting to send the next packet, or just stay on the last setting used. The other approach to handle the effects of stall traffic, is to create additional small background traffic, so that CSI can be updated regularly. Discussion on the pros and cons of each method would be too extensive and goes out of the scope of this chapter. In addition, we are focusing on streaming applications, which provide steady data stream, so in normal operation the CSI never becomes outdated. Our algorithm makes use of the first approach, i.e. it uses timeouts and other safety mechanisms.

3.3 Types of CSI-based rate control

The IEEE 802.11 standard [IEE11] and its supplements do not specify an algorithm for automatic rate selection. So those are the companies manufacturing 802.11 interfaces, that have to come up with the algorithm: they have the freedom to design and implement their own proprietary schemes for control. To our knowledge, all of the known vendors of 802.11 equipment use statistics-based approaches for rate control. The control algorithms can be implemented in software as part of the device driver for the wireless NIC, but also as part of the chipset, which allows for precise control of individual frame (re-)transmissions. In the research community, another class of rate control algorithms has been studied. These control algorithms use SNR-related information as a feedback to improve the sensitivity to changes in link conditions. Up till now, however, such algorithms were not implemented in practical systems, and only simulation results were reported.

In the discussion below we will describe the main representatives of both classes (statistics- and SNR-based) of rate control algorithms. It should be noted that the performance of all algorithms can be improved by differentiating to packet length, because the probability of a packet being corrupted depends on the length of the transmission. Consequently, long packets should be transmitted at a lower rate than short packets.

3.3.1 Statistics-based automatic rate control

An easy way to obtain the necessary information on the link conditions is to maintain statistics about the transmitted data like the frame error rate (FER), acknowledged transmissions, and the achieved throughput. Since these statistics are directly related to the effective user-level data throughput, they inherently guarantee that this throughput is maximized on the long-term. These factors (simplicity and stability) explain the dominance of statistics-based feedback in current 802.11 products. Three basic types of statistics-based rate control can be distinguished: throughput-based, FER-based, and retry-based rate control. The throughput-based approach is the one that uses the most global type of statistic and is the slowest method. The retry-based control uses the most local statistic (number of retries per frame), and is the fastest method. Each statistics-based rate control type is briefly discussed in the rest of this subsection.

Throughput-based rate control

In this approach, a constant small fraction (10%) of the data is sent at the two adjacent rates to the current one (an adjacent rate is the next higher or lower one available). Then, at the end of a specified decision window, the performance of all three rates is determined by dividing the number of bytes transmitted at each rate by their cumulative transmission times. Finally, a switch is made to the rate that provided the highest throughput during the decision window. Atheros uses this algorithm in the NIC driver that they provide for their 802.11a products based on the AR5000 chipset.

To collect meaningful statistics, the decision window has to be quite large (i.e., about one second). On the one hand this makes the algorithm resilient to short-lived changes in the link quality caused by, for example, Rayleigh fading. On the other hand, it prevents swift reactions to long-lived changes in link conditions, which noticeably affects the real-time performance of streaming applications.

FER-based rate control

In this approach, the Frame Error Rate (FER) of the data stream transmitted over the link is used to select an appropriate rate. The FER can easily be determined since under 802.11, all successfully received data frames are explicitly acknowledged by sending an ACK frame to the sender; hence, a missing ACK is a strong indication of a lost data frame. By counting the number of received ACK frames and the number of transmitted data frames during a rather short time window, the FER can be computed as the ratio of the two.

The FER can be used to select the rate setting for the next time window as follows [Bra01]:

downscaling If the FER exceeds some threshold and the current rate is not the minimal rate, then switch to the next lower rate.

upscaling If the FER is close to zero (i.e., below a second threshold), probe the link at the adjacent higher rate with a few (usually even only 1) frames. If all of them get acknowledged, switch to that rate. To prevent the control algorithm from oscillating between two adjacent rates, the upscale action may be prohibited for some time after a downscale decision.

The width of the time window and the thresholds mentioned above are critical for the performance of the FER-based algorithm. The optimal settings of the parameters are dependent on the link and the application, but are generally fixed at design time. Again, this hampers the performance of streaming applications, since a time window tuned for quick responses of typical download applications (to changes in link conditions) yields unreliable FER statistics at low traffic rates. Hence, many frames are transmitted at a non-optimal rate.

Retry-based rate control

An improvement over the FER-based approach is to downscale immediately when the MAC is struggling to transmit a frame correctly over the link. That is, to select the next lower rate after a small number of unsuccessful retransmissions (usually 5-10 retries) [Kam97; Veg02]. This approach is implemented in hardware, as precise control of the rate setting in between retransmissions (of the same frame) is required.

The advantage of the retry-based approach is that it combines a very short response time (a few frames) for handling deteriorating link conditions (downscaling) with a low sensitivity to traffic rates. The price to be paid is that the control algorithm is rather pessimistic. Relatively short error bursts cause long drops in throughput because upscaling to higher rates takes much longer than downscaling due to the need to collect a meaningful FER and to prevent oscillation.

One other important disadvantage of the retry-based approach is that in case of collisions (when other stations are trying to transmit simultaneously), the algorithm will fall back because of the increase of the retries per frame. First, this will cause an undesired drop in throughput (because we switch to a lower rate), which in fact will add on the loss of the throughput already caused by the contention for the medium. Second, unnecessary fallbacks to low rates cause unfairness to the other users as well, because the additional air-time reduces their throughput (see [Heu03]). Unfortunately, without using additional CSI feedback, there is no way that the control algorithm can distinguish between different causes for a frame being retransmitted (bad link or collisions). Therefore it can not avoid unnecessary switching to lower rates in case of medium contention.

3.3.2 SNR-based automatic rate control

A fundamental limit of indirect, statistics-based feedback is that it classifies link conditions as either "good" or "bad". This binary information provides some notion about the direction in which to adapt the rate setting, but does not suffice to select the appropriate rate at once. This leads to a slow step-by-step accommodation to large changes in conditions, and introduces the risk of oscillation in stable conditions. A better approach is to use direct measurements of the link conditions.

The Signal-to-Noise Ratio (SNR) is directly related to the bit-error rate in the link and, hence, to the FER. Consequently, the SNR is linked to the packet delay and jitter, and the throughput, and holds the potential of providing rich feedback for automatic rate control [Bal99]. Knowing the current SNR and the throughput-vs-SNR curves for each rate setting (e.g., Figure 3.1) solves the rate-selection problem instantly: we simply switch to the rate with the highest throughput for the current SNR. This can be implemented efficiently by means of a lookup table.

Despite the advantages, SNR-based rate control has not been applied in practice so far. This is mainly because of three reasons. First, in reality, for certain link conditions the relation between the optimal rate and SNR is highly variable. This is due to the imperfectness of the models describing the radio channel, and also because the link quality depends to some extent on other parameters as well. Second, it is not trivial to obtain a reliable estimate of the SNR of a link. Many radio interfaces provide only an uncalibrated Signal Strength Indication (SSI). Third, the rate controller, which is at the sending side, needs in fact the SNR observed at the receiving side. The problem of communicating the SNR information back is addressed in the emerging 802.11h standard [IEE11h], but the standard has not been finalized yet. In any case, this will be always an open issue for the products that do not support this standard. Also, being aimed at 5GHz, it is not possible for 802.11h to be a successor for the standards like 802.11b and 802.11g that operate at 2.4GHz. This would require defining a new supplement. Another drawback of 802.11h is that the SNR-related information (referred as Link Margin there) is transmitted back by the help of an additional management frame. The latter will increase the MAC overhead, and can cause the SNR information to be delayed due to contention for the medium, and thus not delivered on time.

Most work on using SNR information for automatic rate control is based on simulation and does not consider the practical difficulties of obtaining good SNR estimates. It concentrates on the way in which the noisy and drifting SNR (problem 1) can be used to determine the correct rate setting [Bal99; Ue98]. Holland et al. [Hol01] do address the issue of how to communicate back SNR values (problem 3), but their rate selection algorithm still relies on a straight SNR threshold technique. Another approach is discussed in [PP03], where the assumption is made that the channel is symmetric, meaning that the SNR observed at either station is very similar for any given point in time. This assumption allows Pavon et al. to use the SNR of the last ACK frame as an indicator of the SNR at the other side, and to use it for selecting the rate of the next data frame to be sent. Pavon et al. avoid the issue of estimating the true SNR out of an SSI reading (problem 2) by continuously adapting a table that maps SSI into throughput for the four 802.11b data rates. The adaptation is necessary to accommodate for varying noise levels. However, their control algorithm does not utilize the full potential of SNR information, since the adaptation only takes place on retransmissions; hence, adapting upwards is not supported well. Also, they provide simulation results only; therefore it is not clear how their algorithm will behave in practice. Finally, they do not discuss the dynamics of their approach, that is, how fast the rate setting reaches the optimal value for certain conditions, after these conditions were established.

3.3.3 Hybrid automatic rate control

Both the statistics-based and the SNR-based approaches have their advantages and disadvantages. The statistics-based approach gives robust performance and inherently maximizes the throughput in the long term. However, the main drawback is its slow response to changing link conditions, which can be a source of problems for real-time applications. The SNR-based rate control can respond very fast, but due to the uncertain and fluctuating relation between SNR information and BER of the link, it lacks stability and reliability. Therefore, a logical step forward is to combine the two approaches in a hybrid algorithm that will provide both robustness and fast response.

This chapter describes such an algorithm and its implementation.

3.4 Practical implementation of hybrid CSI rate control

In this section we describe an advanced rate control algorithm that combines the advantages of statistics-based and SNR-based methods. The goal is to support streaming applications by limiting the packet delay and jitter as much as possible, even at the expense of throughput when necessary. This requires a hybrid

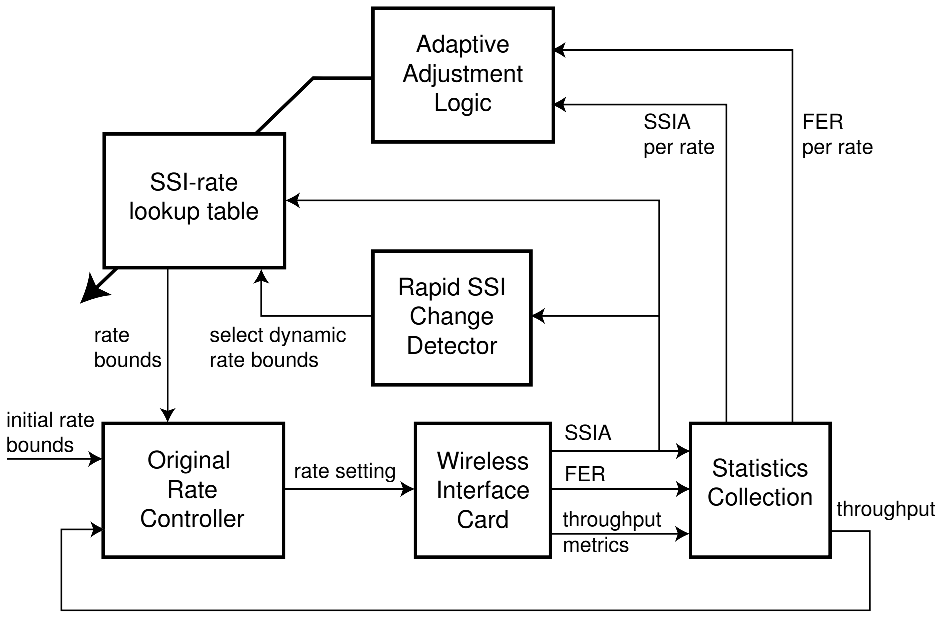


Figure 3.2: *Structure of the hybrid rate controller.*

approach, since under stable link conditions we want to provide a robust, high-throughput connection (statistics-based approach), but under volatile conditions induced by, for example, user movement, we want to avoid retransmissions by rapidly switching to a lower rate (SNR-based approach). Another important goal is to design a working system, so we must address the practical problems associated with SNR-based methods as discussed in the previous section. Results of our experimental prototype will be presented in Section 3.5.

The core of our hybrid algorithm is a traditional statistics-based controller. More specifically, it is a throughput-based rate controller, which probes adjacent rates to determine if a rate switch is necessary (see Section 3.3.1). Figure 3.2 shows the complete structure of our hybrid rate controller; the throughput-based controller is part of the feedback loop depicted at the bottom of Figure 3.2, including the Wireless Interface Card and Statistics Collection component.

The decisions of the core controller can be overridden by a second feedback loop. This loop bounds the acceptable range of the Signal Strength Indication of the Acknowledged frames (SSIA) values for each rate, based on the specific knee in the Throughput-vs-SNR curve (cf. Figure 3.1). These SSIA bounds are implemented as a lookup table (SSI-rate lookup table component in Figure 3.2), which is indexed by the intended rate, given by the core controller. Such a table is presented in Table 3.1.

For example, if the core controller decides on a rate setting of 36 Mbit/s, and at the same time the last measured SSIA is 12 dB, then the rate that will be actually used is 12 Mbit/s. The bounds on the maximum rate are tightened

Rate (Mbit/s)		6	9	12	18	24	36	48	54
Low threshold (dB)	stable	7	9	11	13	15	18	22	25
	volatile	12	14	16	18	20	23	27	30
High threshold (dB)		17	19	21	23	25	28	32	35

Table 3.1: *Sample SSI-rate lookup table*

when channel conditions change rapidly. This is done by addressing the threshold values marked as “volatile” in the SSI-rate lookup table, instead of the “stable” values that are used in slow-changing channel conditions. The rationale of this scheme is that it is better to play safe in volatile conditions, and switch to a lower rate to minimize packet jitter by reducing the probability of a frame loss. Rapid changes in channel conditions are detected by sensing the changes of consecutive SSIA readings; the component is called Rapid SSI Change Detector (RSCD) and is shown in the middle of Figure 3.2. To account for the drift of the SSIA readings, we employ an adaptive adjustment logic that updates the values in the lookup table according to the recent history of channel conditions. This approach is similar to that described by Pavon et al. in [PP03], and may be regarded as an auto-calibration loop.

The operation of the hybrid control algorithm will be detailed further in two parts. First, we will describe the extended rate controller, which comprises all components in Figure 3.2 except the adaptive adjustment logic. Second, we will discuss the auto-calibration loop, that is, the details of the adaptive adjustment logic.

The SNR-based extension of the rate controller is centered around the SSI-rate lookup table. This table contains three SSI thresholds per rate: two for stable conditions (high and low thresholds), and an additional low threshold for volatile link conditions. These thresholds describe for each rate setting the SSIA values that are allowed for its use. The low threshold used in stable conditions is referred to as the *absolute* low SSIA value that provides acceptable performance (defined as FER $< 10\%$) under ideal circumstances. The low threshold used under volatile conditions is referred to as the *dynamic* low SSIA value. The RSCD decides which of the two low thresholds (absolute or dynamic) will be used for bounding the rate selection for the next frame. The detector observes the differences between three consecutive SSIA readings, comparing them only if they are taken within a certain, limited time period. If both differences have the same sign, and their sum exceeds a certain threshold, then the dynamic low threshold is used, since conditions are changing fast and consistently. Otherwise, the absolute low threshold for stable conditions is used. The RSCD has also a hold property, meaning that it will keep the indication for rapid change active for certain small amount of time, even if this change is no longer detected. This feature improves the stability of the controller and is based on the assumption that link quality changes are caused by physical processes, which require at least some minimal time to take place.

The selection of the rate `RS_curr` of the next frame to be transmitted proceeds as follows:

```
once_per_decision_window() {
    RS_opt = find_opt_rate_by_stats();
    may_upscale = true;
}

for_each_packet() {
    RS_curr = probe_or_not(RS_opt);
    calculate_bounds();
    if (RS_curr > RUpBound)
        RS_curr = RUpBound;
    else if (RS_curr < RLoBound &&
                        may_upscale) {
        RS_curr = RLoBound;
        try_upscale = true;
    }
    xmit_success = xmit_packet(RS_curr);
    if (xmit_success)
      update_SSIA_stats();
    if (try_upscale) {
        if (xmit_success)
            RS_opt = RS_curr;
        else
            may_upscale = false;
        try_upscale = false;
    }
}
```

Here, `RS_opt` is the optimal rate as calculated by the core statistics-based algorithm at the end of each decision window. The upper and the lower rate bounds (`RUpBound` and `RLoBound`) are calculated from the SSIA thresholds in the lookup table and the SSIA from the previous packet (`prev_SSIA`), by the `calculate_bounds` procedure listed below. If the transmission is successful (indicated by the flag `xmit_success`), we also update the SSIA statistics by the `update_SSIA_stats` procedure. Note that when the selected rate (`RS_curr`) falls below the lower bound (`RLoBound`), we attempt to upscale by transmitting a frame at the `RLoBound` rate. If the transmission succeeds, we continue to use that rate. If it fails, we block additional attempts for the remainder of the decision window. This behavior is achieved by the help of the flags `may_upscale` and `try_upscale`.

```
calculate_bounds() {
    for (i=max_rate_index, i > 0, i--) {
        if (prev_SSIA >=
                SSIA_tbl[i].lo_thld[is_dyn])
            break;
    }
    RUpBound = RateTable[i];
    for (i=0, i < max_rate_index, i++) {
        if (prev_SSIA <= SSIA_tbl[i].hi_thld)
            break;
    }
    RLoBound = RateTable[i];
}
```

The low SSIA thresholds determine the upper rate bound for the current SSIA value. Likewise, the high SSIA thresholds determine the lower rate bound. When there are rapid changes in channel conditions, the SSIA dynamics detector raises the `is_dyn` flag above, causing the algorithm to use the dynamic SSIA thresholds for calculating a stronger `RUpBound`.

Before we explore the work of the adjustment logic, lets first discuss how we can detect the situations when the SSIA thresholds need to be adjusted. Normally, a SSIA threshold should be positioned near to the right of the knee of the FER-SSI curve for the corresponding rate setting. Lets suppose that the threshold for rate k is too much to the left. Then there will be packets, attempted at k, and with SSIA greater then the SSIA threshold, which will experience many retransmissions before they are successfully received at the other side. This can be used as an indication that the SSIA threshold for rate k should be increased. On the other hand, if the SSIA threshold for rate k is too much to the right, then almost all packets, allowed for transmission at rate k, will be sent successfully at the first transmission attempt. So, this would be our indication that the SSIA threshold for rate k should be decreased.

For each rate, two counters are kept, which are updated on per-packet basis. The first counter, `lo_SSIA`, keeps the number of the packets, whose transmission resulted in indication that the threshold for this rate should be increased. Similarly, the second counter, `hi_SSIA`, keeps the number of the packets, whose transmission resulted in indication that the threshold for this rate should be decreased.

At the end of each decision window, the update of the thresholds for all rates that were used in this decision window, is performed as follows:

```
Adjust_SSIA_thresholds() {
    if (#packets < MIN_PKT_NUM)
        return;
    for (i=0, i < max_rate_index, i++) {
            if (SSIA_tbl[i].lo_SSIA / #packets > a)
                SSIA_tbl[i].lo_thld += c;
            else if (SSIA_tbl[i].hi_SSIA / #packets > b)
                SSIA_tbl[i].lo_thld -= d;
            SSIA_tbl[i].hi_thld = SSIA_tbl[i].lo_thld + e;
            SSIA_tbl[i].lo_thld_dyn = SSIA_tbl[i].lo_thld + f;
        }
    }
```

Here, a, b, c, and d are coefficients that determine the accommodation properties of the adaptation algorithm. Coefficients a and b are thresholds, used together with `lo_SSIA` and `hi_SSIA` as an integrating detection mechanism to determine the need and the direction of SSIA threshold adjustment. Thus, a and b determine the filtering properties of this detection mechanism. Coefficients c and d determine the magnitude of a single adjustment to the SSIA thresholds per decision window, in the event we detect that it is necessary to make an adjustment during this window. These coefficients can be compared to the D (differential) component in a PID controller. In the current implementation, e and f are constants, but in the future, they will represent functions. The e function accounts for the magnitude of the drift in the FER-SSI relations. The f function determines how pessimistic the algorithm will be in the case of rapidly changing channel conditions, thus influencing the responsiveness of the rate controller.

Tuning the coefficients is done by experimenting with different sets of values, and selecting the ones that produce the best performance results, yet yield a stable operation under all operating conditions that we have tested. Initial analysis shows that the algorithm is most sensitive to the choice of a, b thresholds, but fortunately determining their values is subject to the least variance in our experiments. Currently, the values used are $a = 0.1$, $b = 0.8$, $c = 1$, $d = 1$, $e = 10$ and $f = 5$. Part of our future work is to perform a thorough sensitivity analysis, and to establish a firm, model-based procedure to adjust the coefficients, depending on the particular case the hybrid algorithm is used for.

After updating the thresholds for the rates used during the decision window, the thresholds for the unused rates may need to be updated as well to preserve the monotonic value growth in the lookup table. An example of a specific update situation is shown in Figure 3.3, where the thresholds for rate indexes 1 and 4 have already been updated according to the procedure outlined above. Next, the thresholds for the other rate indexes are checked. Rate index 2 is found to violate the monotonicity requirement, and its value is increased to that of the preceding rate index 1. No further adjustments are needed. To handle potential conflicts between threshold updates, the adjustments are performed from the lowest rate to the highest rate.

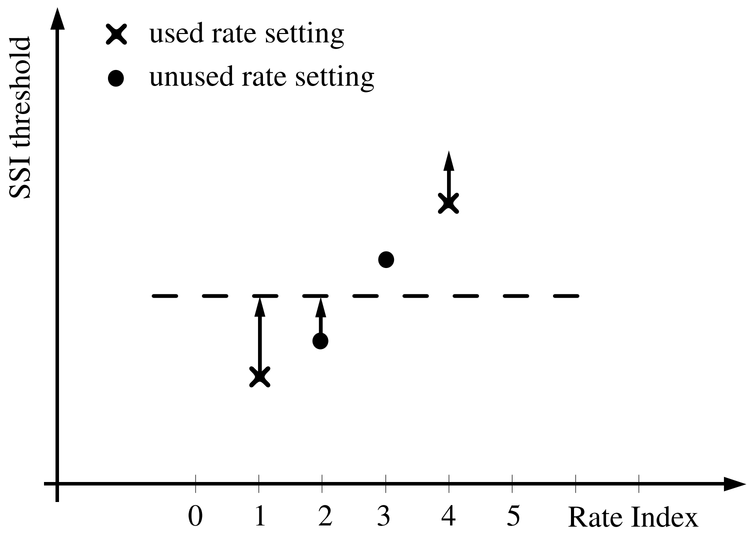


Figure 3.3: *Resolving specific cases for SSIA thresholds update.*

3.5 Experimental evaluation

We have implemented our hybrid algorithm on real hardware and carried out a number of experiments executed in real-time. In this section, the traffic we use is synthetic, and the main goal is to explore different parts of the algorithm, and to give assessment for their contribution to the performance. A real scenario evaluation will be made in the next section.

3.5.1 Experimental setup

Our experimental setup consists of two laptops, both equipped with a Proxim Skyline 802.11a wireless network interface card supporting eight different data rates, ranging from 6 to 54 MB/s. These cards are based on the Atheros 802.11a AR5000 chipset. The advantage of using this chipset is that it only implements the lowest MAC functionality (e.g., basic medium access, retransmissions control, etc.) and leaves the remainder to the device driver. This provides us with the flexibility to modify the original MAC protocol, and even to replace the automatic rate controller with our own hybrid approach.

The laptops are running Linux, and we developed the AR5000 device driver ourselves since only a Windows version was supported by Atheros. The device driver includes our hybrid automatic rate controller discussed in the previous section. For experimentation purposes the driver can be instructed to disable the SNR-based components so that we could measure the performance of the original throughput-based controller developed by Atheros. We modified the popular Netperf 2.2 tool to set the low-level MAC parameters exported by our device driver: maximum number of retransmissions, rate setting, Tx power, etc. We also used Netperf to generate data traffic and to collect statistics about the raw performance of the wireless connection between the two laptops in a standard office environment (with concrete walls).

3.5.2 SSIA reasoning

To obtain basic insight into the performance of the wireless network cards, we conducted a first experiment in which we placed one laptop at different positions in the corridor. At each position we exercised all eight 802.11a rate settings logging the throughput reported by Netperf and the signal strength (of the acknowledgements) as reported by the network interface. Figure 3.4 shows the relationship between signal strength, rate setting (modulation scheme), and throughput for our experimental setup. In addition to the raw measurements, Figure 3.4 includes a smoothed[5] curve for each rate setting to better visualize the cliff behavior. The infeasibility of directly implementing a SNR-based rate control algorithm can be seen for example by looking at the results for 24 Mbit/s. Note that the data points represent averaged values over time. But still, some of the points (like those at an SSIA of 23 with throughput below 2Mbits/s) suggest that the cliff for that rate should be at about a SSIA of 23, while others on the left determine the switch over point to be around 15. Thus, straightforward SNR-based control will lead to unstable, and sometimes erratic behavior.

The SSI readings that we obtained from the chipset ranged in practice from 0 (complete loss of connection) to about 70 (cards next to each other). Unfortunately, the documentation about the AR5000 chipset does not specify the unit of these measurements. Therefore we performed a series of open-space experiments in which we varied the distance, which revealed that the obtained SSI values are in dB over some reference point. This reference point is most likely linked to the noise floor of the receiver, thus we end up getting 'SNR-related' information. Our experiments show that the noise floor of the receiver is recalibrated periodically, which contributes to the drifting in time of the throughput-SSI curves. It is important to note here that, since we use a self-adapting algorithm, the reference point of the SSI readings does not influence the operation of the algorithm. This is provided, of course, that the reference point does not change too fast (within a second), which is the case in our experimental setup.

Figure 3.4 clearly shows that in reality, the obtained SSI feedback is quite noisy, especially around the edge of each cliff. In addition, the noise level for more complex modulation schemes (i.e., higher data rates) is quite high, even on the flat part of the curve. For example, at an SSI of about 42, the throughput of the three top rates suddenly plummets to a rate of just 1 Mbit/s. This is most

[5]To smooth the data, we first excluded values that violate the monotonic increase of the curves. Then, we interpolated the data to obtain regularly spaced values. Finally, we ran the data through a low-pass FIR filter.

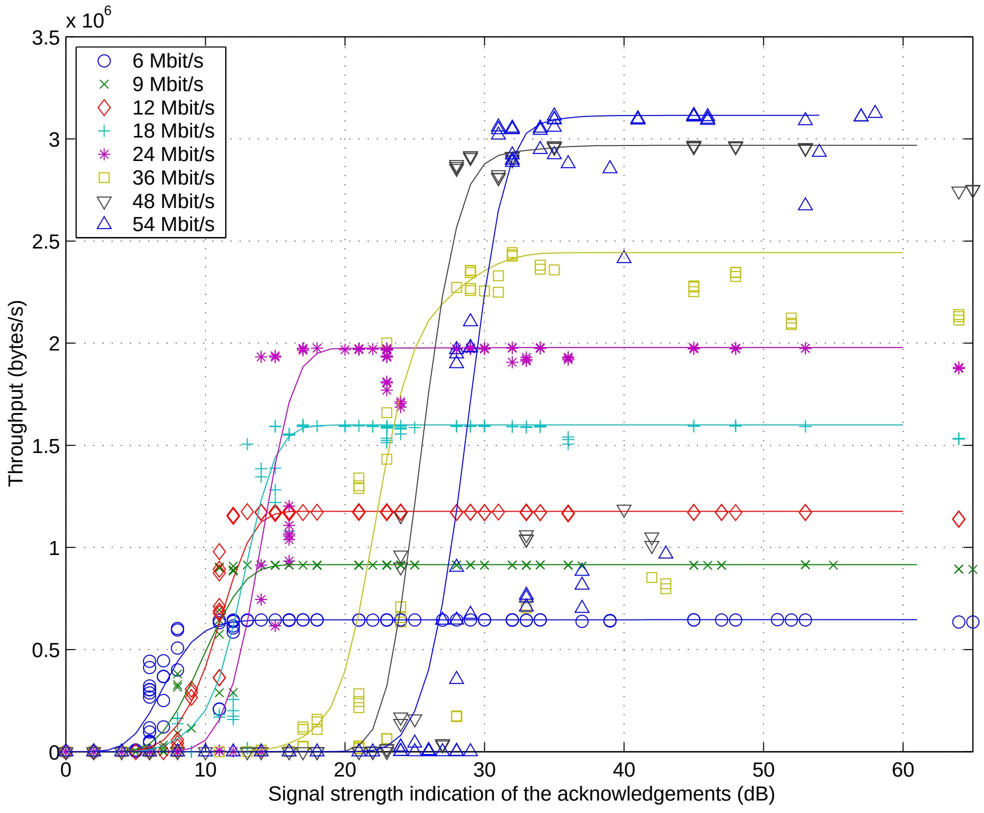


Figure 3.4: *Throughput vs. SSI of acknowledgments (SSIA).*

probably due to complete Rayleigh fading of some of the OFDM carriers used in the 802.11a radio at that particular position in the corridor.

A second observation that can be made from the curves in Figure 3.4 is that, in our experimental setup, it does not make sense to use the 9 Mbit/s rate setting at all, since this setting does not outperform any other rate setting for any SSIA value. Although we did not investigate if this holds for any possible scenario, all our experiments, and also simulations by [Qia02] show that the 9 Mbit/s rate setting is generally of no use.

Finally we like to point out the throughput reduction caused by MAC and PHY layers overhead. This overhead is due to extra information added to the data payload at MAC and PHY layers (in the form of headers, preambles, etc.), and the fact that the information is transmitted at a fixed (low) rate. This overhead is part of the 802.11 standard [IEE11; IEE11a], and can cause significant deviation from the PHY raw data throughput [Jun03], especially at high rate settings. At the 6 Mbit/s setting, we logged a Netperf throughput of 4.9 Mbit/s, hence, the MAC/PHY overhead is limited to 18%. At the 54 Mbit/s setting, the Netperf throughput is limited to just 23.8 Mbit/s, hence, the overhead has grown to 56%. Note that the actual throughput, and consequently the MAC/PHY overhead, depends on the frame size used. In our case, the packets are 1024 bytes long, and we did not use packet fragmentation, so the frame size is equal to the packet size.

We conducted a second experiment to verify the assumption that the communication channel is symmetric, so that we can use the signal strength of the acknowledgements (SSIA) observed at the sender instead of relaying the true signal strength (RSSI) observed at the receiver. The experiment involved a person walking around in a random fashion with one of the laptops (the other laptop remained stationary), and recording both the SSIA at the transmitting side and the SSI at the receiving side for each packet. Figure 3.5 shows how the signal strength observed at the receiver changed during the 140 sec experiment (packets are streamed at a rate of 100 per second). Note that during stable conditions (standing still), the noise in the SSI is limited to ± 2 dB; while moving, however, the fluctuations are much larger (up to ± 17 dB).

Figure 3.6 provides a histogram of the differences between the signal strength observed at the sender (SSIA) and receiver (RSSI). Even though the SSIA and RSSI readings are not captured from the channel at the same time (the ACK follows the DATA packet), it is quite clear that the SSIA and RSSI values are strongly correlated. For 74% of the packets, the difference falls within the stationary ± 2 dB noise range, and even the largest differences are in line with the noise during movement. Therefore, from a control perspective, we can safely use SSIA instead of RSSI, thus avoiding the problem of relaying the SSI at the receiver back to the rate controller operating at the sender.

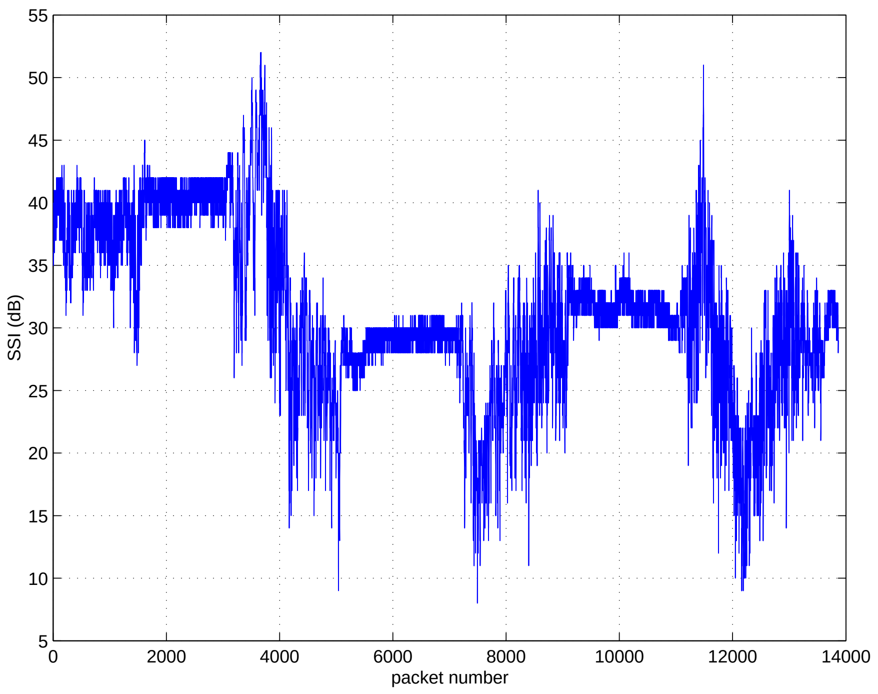


Figure 3.5: *SSI at the receiving side (RSSI), random walk.*

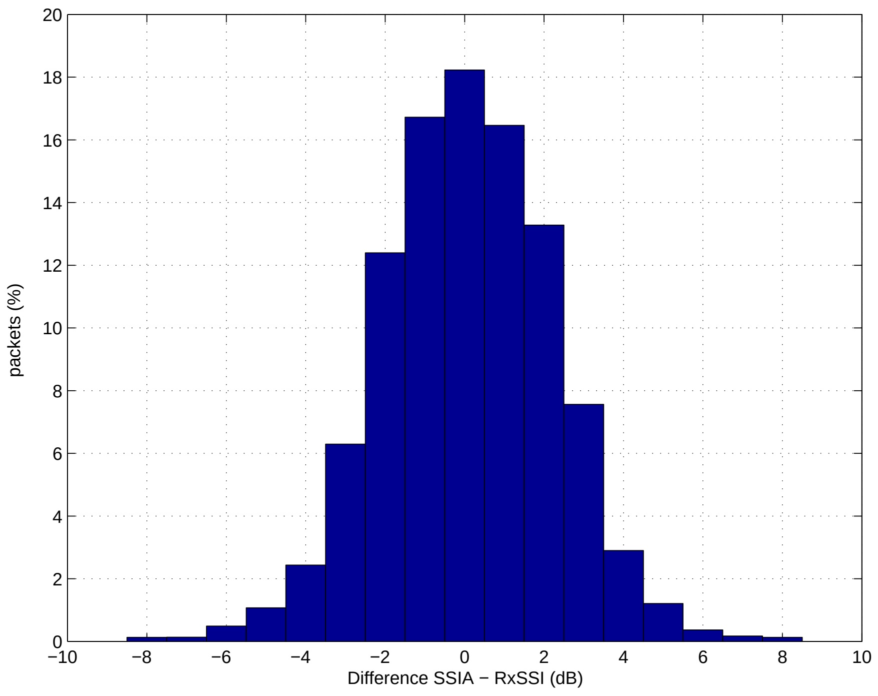


Figure 3.6: *Histogram of the difference SSIA - RSSI.*

3.5.3 Step response functions

In the third experiment we compared the original throughput-based rate controller supplied by the manufacturer of the chipset (Atheros) with our hybrid controller. To allow for a head-to-head comparison under the same conditions, we decided not to use a real-life scenario with a walking person, since repeating the experiment (i.e., regenerating the exact same channel conditions) would be close to impossible. Instead, we measure the Step Response Function (SRF) in an artificial setting to determine the responsiveness, robustness, etc. of the rate control algorithms.

In general the SRF is used to study how some control system reacts when a certain input parameter (in our case, the SSI) changes instantly, that is, in one step from one value to another. In our SRF experiment we placed one of the laptops in a microwave oven, started streaming data to the other laptop, and then closed the door rapidly, causing the SSI to swiftly drop from about 35 (good) to about 10 (bad), because the shielding of the microwave oven blocks almost all radio waves in the 5 GHz band. After about 3 seconds the door was opened again, generating a second step in the channel conditions (from bad to good). Since the microwave door was opened and closed manually, the exact channel conditions (i.e, SSI readings) show small variations between experiments (cf. figures 3.7 and 3.8). Nevertheless, we can reproduce the SSI steps with enough accuracy to show the (large) differences between the throughput-based and hybrid rate control algorithms. To stress these differences we impose a traffic load that corresponds to a demanding streaming-video application; Netperf was instructed to send 1024 bytes long packets at a rate of 100 packets/s, resulting in an effective data rate of 800 Kbit/s.

Figures 3.7, 3.9, and 3.11 show the SSIA readings (input), generated rate settings (output), and induced packet latencies (performance), respectively for the original throughput-based rate control algorithm. Figures 3.8, 3.10, and 3.12 show the same for our hybrid algorithm. According to expectations the throughput-based controller cannot accommodate fast enough to the step down in channel conditions (around packet number 1000) because of the 1 second decision window. When it does respond (around packet number 1080), the controller decides to step down one rate setting (from 54 to 48 Mbit/s), which is too little too late. All packets transmitted during the closing of the microwave get lost; the latencies for these packets are labelled inf(inity) in Figure 3.11. As a consequence, no acknowledgements are received, causing a gap in the SSIA readings shown in Figure 3.7 and, hence, a lack of throughput statistics leading the controller to maintain the current (way too high) setting. This situation continues until the channel conditions improve after opening the microwave door, and the controller can finally decide to step down further (from 48 to 36 Mbit/s) around packet number 1380. Our hybrid controller, in contrast, responds almost immediately to the drop in channel conditions and overrules the decisions of its internal throughput-based controller based on the SSIA thresholds. Consequently, almost all packets are transmitted at the appropriate rate and service continues smoothly. When the conditions improve (around packet number 1290) the hybrid controller is also quick to respond and switches to higher rate settings.

The step-response function in Figure 3.9 shows the workings of the throughput-based controller in quite some detail. First of all, we can clearly see the 1 second decision window in the rate selections, for example, packets numbered 1380 to 1680 are grouped into three windows with rates 36, 24, and 36 Mbit/s respectively. Also, the probing at adjacent levels (1 in 10 packets) is visible as the light bars up and/or below the dark bar of the selected rate. Figure 3.10 shows that the hybrid algorithm does probe at adjacent levels too, but that quite often the SSIA-based thresholds overrule an up-probe. As an example, consider the time frame from packet number 1500 to 1750. In this period, the majority of the packets is sent at the 48 Mbit/s rate, some are sent at the adjacent lower rate (36 Mbit/s), but none at the higher rate (54 Mbit/s). This behavior follows from the data in Figure 3.4, which shows that around 35 dB the 54 Mbit/s throughput curve is still rising, signalling unstable conditions.

The packet latency, as reported in figures 3.11 and 3.12, is measured as the time between the moment the device driver inserts a packet in the send queue, and the moment that the chipset starts the actual, physical transmission. The latter time stamp is returned (by means of an interrupt) to the device driver when the acknowledgement packet is received. The chipset automatically takes care of retransmissions. If the maximum number of retries (10 in our case) does not result in a successful transmission, the packet is dropped, and an interrupt is generated to inform the driver. The device driver then marks the latency for the dropped packet as infinite. Under normal conditions the send queue is (almost) empty, but when retransmissions are issued, the number of packets in the queue grows. When the control algorithm then switches to the “right” rate, the packets in the queue are transmitted back to back until the queue becomes empty again. This effect is clearly visible in the latency graphs: after a sequence of dropped packets, latency jumps and then rapidly declines to (near) zero. Streaming applications are quite sensitive to latency and usually put a strict requirement on the maximum latency for any packet.

The bottom line is that, from the point of view of a streaming application, the difference between the two automatic rate controllers is huge. With the original throughput-based controller, the stream is halted for about 2 seconds (205 packets are lost), causing a complete outage in service to the user. With the hybrid controller, the user may experience some service degradation depending on the application’s ability to cover up the few (5) dropped packets. In the case of a streaming video application, the difference is between losing connection and observing some artifacts in a few frames; a significant difference indeed.

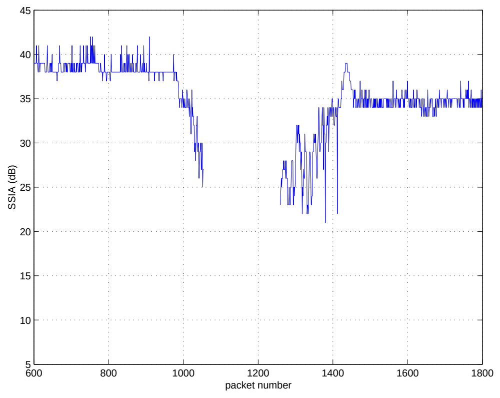


Figure 3.7: *SSIA for the statistics-based algorithm.*

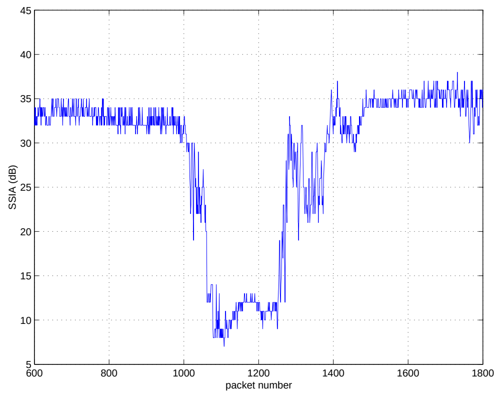


Figure 3.8: *SSIA for the hybrid algorithm.*

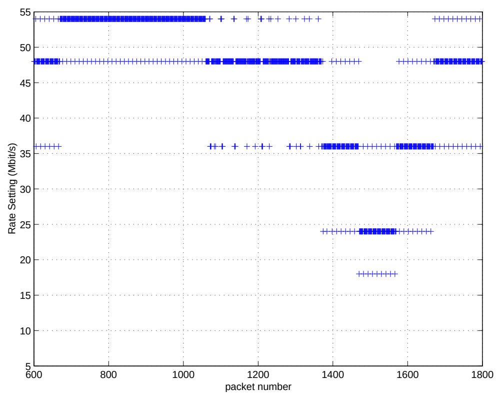


Figure 3.9: *Step response function - statistics-based algorithm.*

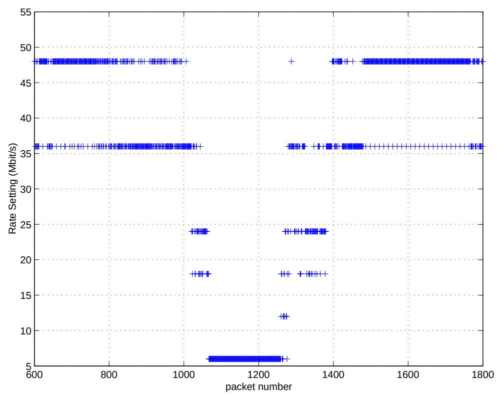


Figure 3.10: *Step response function - hybrid algorithm.*

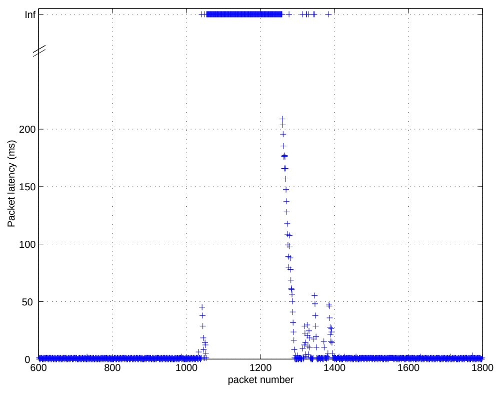


Figure 3.11: *Packet latency - statistics-based algorithm.*

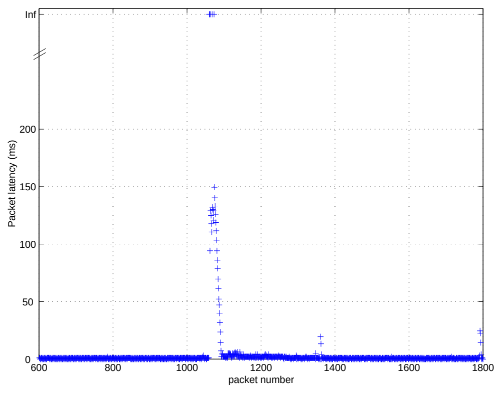


Figure 3.12: *Packet latency - hybrid algorithm.*

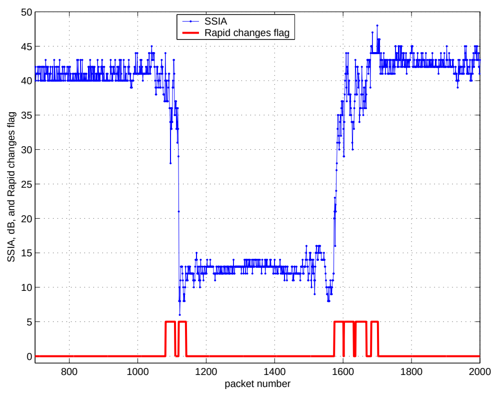


Figure 3.13: *Dynamic detection algorithm performance.*

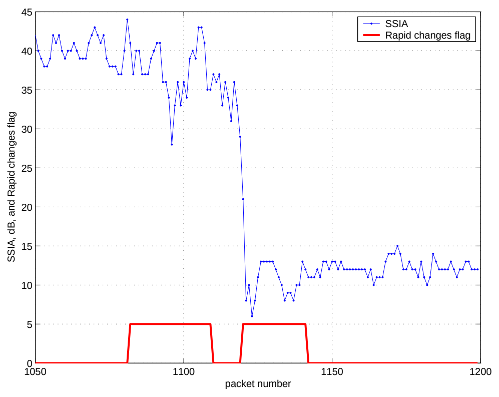


Figure 3.14: *Dynamic detection algorithm performance – zoomed.*

3.5.4 Rapid SSI Change Detector (RSCD)

We expect that delays and packet losses will be quite critical for multimedia applications (leading to artifacts in a video and dropouts/hickups in video/audio, see Section 3.6), therefore it is very important that we avoid delays and losses as much as possible. As it was already discussed, RSCD detects the moments when the link conditions change drastically. Then it instructs the rate controller to become more conservative, effectively providing the necessary safety margin in time, in case the algorithm has to react to rapidly declining link conditions. Here, we evaluate the performance gain that this detection circuit adds, by comparing the results of the work of the rate controller with, and without, RSCD.

Figures 3.13 and 3.14 show the moments that RSCD determines as "turbulent", i.e. those in which there is a rapid change in the link conditions.

By comparing different situations, we can tune the parameters of RSCD so that it is sensitive enough to identify the really turbulent periods, and in the same time, not to produce (too many) false alarms.

The results for the comparison of the performance of the rate control systems with, and without RSCD, is shown in Figures 3.15 and 3.16.

The benefit of using RSCD is clearly seen, as the peak latency for the system with RSCD barely reaches 100 ms, while the peak latency for the system without RSCD exceeds 250 ms. Also, with RSCD, the lost packets are 44, opposed to 126 lost packets in the case when RSCD is not used.

3.5.5 SSI thresholds adaptation circuit (STAC)

We have already discussed that STAC is responsible for adjusting the thresholds in the SSI lookup table, following the shifts of the FER-SSI curves. A good way to evaluate the operation of STAC is to set the initial SSI thresholds to some

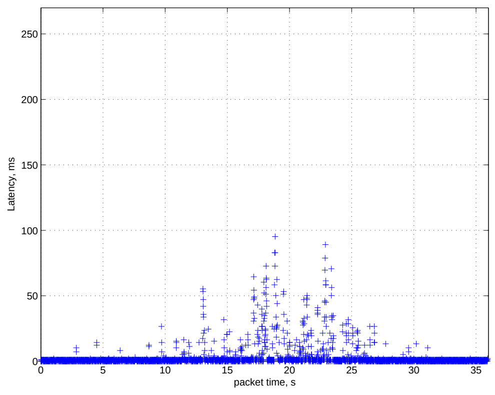


Figure 3.15: *Packet latencies with dynamic detection algorithm.*

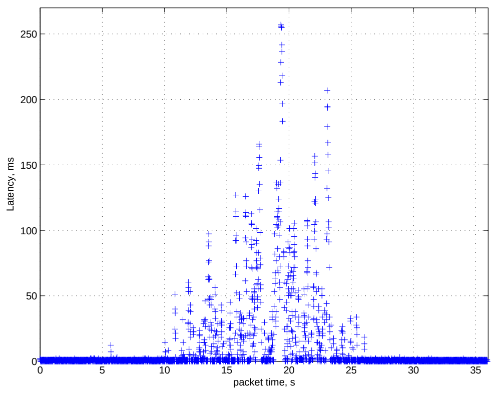


Figure 3.16: *Packet latencies without dynamic detection algorithm.*

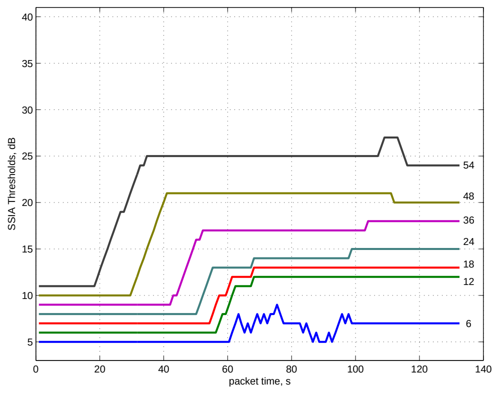


Figure 3.17: *Adaptation of SSIA thresholds with low initial values.*

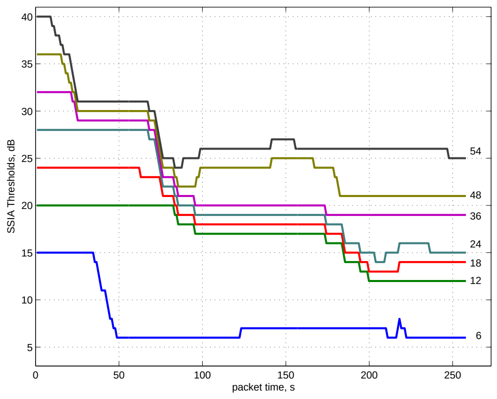


Figure 3.18: *Adaptation of SSIA thresholds with high initial values.*

"invalid" values and observe how STAC modifies them. The results are shown in figures 3.17 and 3.18. The first one shows the adaptation process when the thresholds are set to extremely low values, the second, respectively, shows the accommodation when the thresholds were initially high. As it can be seen, in both cases the thresholds reach values that are close to the average "standard" values, which for the rates from 6 to 54 Mbits/s are 7, 12, 14, 16, 18, 22 and 25, respectively. The average "standard" values were determined by getting the FER-SSI curves under different environmental conditions, and calculating the mean for the SSI values, for which FER was equal to 50% (the middle of the knee of the curves). The curves in figure 3.17 do not reach exactly the same values as these in figure 3.18, because of the drift of the FER-SSI curves in time, and because of noise in the SSIA acquisition.

It can be noticed, that in both cases the full accommodation takes a significant amount of time. However, on the one hand these are extreme cases, and in real situations the initial values would be much closer to the optimal ones. On the other hand, the fluctuation speed of the optimal values is rather low, so STAC will have in practice enough time to accommodate.

The factors that influence the speed of adjustment of the thresholds include the change of the link conditions, the packet rate and burst degree, and many others. Therefore, the discussion about these factors goes out of the scope of this chapter, and will be presented elsewhere.

3.6 Real video-streaming scenario evaluation

In this final experiment we will try to validate the previous experiments that use synthetic traffic, with a realistic video streaming application in an office environment with one of the two laptops moving around.

Because of the nature of video codecs like MPEG1/2 and H.263, lost packets can have disastrous effects on the image quality. To decode a certain (video) frame, the decoder depends on the previous frame being decoded correctly. When lost, a single packet which contains information of part of a video frame, will generate effects on the next frames as well. For any streaming transmission, for example video-conferencing or just television, all packets should arrive in time, otherwise they are basically useless. We assumed a latency constraint of 500 (ms) between transmission and reception. This means we should prevent to lose packets, since there is no time to use end-to-end (TCP-level) retransmissions. The link or MAC-level retransmissions, on the other hand, are an effective way to deal with packet losses.

In this experiment we will compare the decoded video quality for different algorithms. The setup of our experiment is as follows: One of the laptops is on a fixed position on a desk. The other laptop is carried out the room and then up and down through the hallway a few times, and then returns to the initial position next to the other laptop. This experiment takes two minutes. By walking up and down the hallway we ensure that the conditions are continuously improving or deteriorating, so we can inspect the behavior of the rate control algorithm at different circumstances. During the whole experiment the first laptop was streaming video to the second laptop which decoded and recorded the received video. Since we are evaluating a real-time streaming and displaying application, we discarded all packets that arrived when in fact the next video frame should already be decoded.

Each experiment is run three times to be able to correct for slightly varying conditions during the experiments. The transmission consisted of a H.263 encoded stream of the “carphone” sequence at QCIF format. It is a representative video often used for comparing video coders. Since we are just evaluating the effects of the rate-control algorithm on the number of packet losses, the produced video rate is lower than the lowest rate setting (6 Mb/s).

Afterwards, we have investigated the quality of each received stream. A classical and easy measure for image quality is the peak-signal-to-noise-ratio (PSNR) measure. The downside of this measure is that it not necessarily corresponds to the human perception and that it is not well-suited for moving pictures where sometimes frames are missing. To overcome this shortcoming we employed two measures: 1) average PSNR: The average of the PSNR's of all frames. For each missing video frame, the last correctly received frame is taken in place[6]. 2) human perceptual quality. To obtain a perceptual measurement we have shown each received video to fourteen people who had to give a mark between 0 and 5 (0=bad, 5=good), of course without telling to which case this recording belonged. Both measures together should give a good indication of the effects of losing packets on the quality of the video.

We have compared the statistics based algorithm and our hybrid algorithm with a perfect transmission and with a reference algorithm. Table 3.2 shows for the following cases, the packet loss ratio the PSNR quality and the perceptual quality measure on a scale of $0 - 5$.

1. Perfect transmission: In this fictitious algorithm, we simulated that there were no lost packets at all, resulting in the highest quality possible in this setup.

2. Reference algorithm: the rate setting is fixed at the lowest value, therefore with the lowest packet loss probability. This represents the best any rate control algorithm can do under these circumstances.

3. Paranoid Hybrid algorithm: uses the hybrid algorithm, but it switches to lower rates at a slight indication of deteriorating channel conditions, hereby trying to prevent losing packets. This is achieved by shifting upwards the SSIA thresholds in the SSI-rate lookup table. By limiting the number of attempts per decision window, the algorithm is made more conservative to go to higher rates, when the link conditions are good.

4. Hybrid algorithm: our hybrid rate controller as evaluated in the previous sections.

5. Statistics based algorithm.

The results in Table 3.2 show that the statistics based algorithm generated seven times as many packet losses than our hybrid algorithm. The video quality for the hybrid algorithm is therefore 5.4 dB higher than for the statistics based

[6]This also happens visually with a real-time decoder. We did not employ any advanced error concealment techniques at the video decoder, because we want to have a clean comparison between the rate control algorithms

Algorithm	packet loss	perceptual quality	average PSNR (dB)
1. Perfect	0.00%	5.0	37.34
2. Reference	0.08%	4.2	36.96
3. Hybrid – paranoid	0.15%	3.0	36.59
4. Hybrid	0.97%	1.3	34.73
5. Statistics-based	7.01%	0.4	29.33

Table 3.2: *Packet loss and PSNR measurements and perceptual quality rating for five different algorithms*

algorithm. The perceptual rating confirms this difference. As can be seen from the perceptual measurements, even low packet-loss ratios already give significant lower qualities. This is due to the propagation of errors in consecutive frames. Therefore we introduced a more paranoid version of the algorithm. This version goes to greater extent to prevent losing packets by switching to lower rates at an earlier stage and being hesitant to increase the rate when the conditions improve. The result was that fewer packets were dropped, and the achieved quality was closer to the reference algorithm. On the other hand we should remark that on average, this paranoid algorithm will as a result also obtain lower average rates than the normal hybrid algorithm, under the same circumstances. This is nonetheless not visible in our experiments.

3.7 Conclusions and future work

In this chapter we addressed the problem of supporting real-time streaming applications over a wireless link. The strict latency requirements of streaming applications and the fluctuating link conditions require careful handling in all layers of the protocol stack. At the lowest level, the IEEE 802.11 standard includes several tuning parameters, of which the transmission rate (selected per packet) is the most important one. Standard device drivers for today's wireless 802.11 products are based on gathering statistics and, consequently, adapt rather slowly to changes in conditions. Using signal strength (SNR) information is appealing because it provides instant information about the channel conditions. However, since SNR readings are quite noisy, have a wide variance, and are unstable, SNR cannot be used in practice to directly select the rate.

We have designed and implemented a hybrid rate controller that uses the available SNR information as a safeguard. This hybrid controller is built around a traditional statistics-based rate controller. The SNR information is used to limit the range of feasible settings from which the core controller can choose. This prevents the transmission of packets at a too high rate when channel conditions suddenly change for the worse. In a controlled experiment, we measured the step response function of both the statistics-based rate controller provided by the manufacturer (Atheros) of the 802.11 chipset that we employ, and that of our hybrid rate controller. The performance difference is large: whereas the statistics-based algorithm incurs a total dropout due to its slow response, the hybrid algorithm adjusts almost instantly and looses only a few packets. We conclude that SNR information can be used to the advantage despite the practical problems of the noise and drift usually associated with it.

Although the gain of our hybrid algorithm with respect to the statistics based algorithm, is large in terms of packet losses and for instance video quality, the hybrid algorithm can still be improved. By being more paranoid, more packet losses can be prevented. However, there is clearly a tradeoff between the number the packet loss paranoia and bandwidth. We plan to investigate this trade-off more thoroughly.

We also plan to investigate the benefits of including other parameters to be controlled (like packet fragmentation and Tx power). We will also study the possible use of inter-layer QoS negotiations, so that the application will be able to influence the algorithm controlling the radio parameters. Finally, to test our approach in more realistic circumstances, we will investigate more-complex real scenarios with more than two nodes.

Chapter 4

Multi-layer control

The quality of real-time video streaming over wireless links is significantly reduced due to packet losses and delays caused by rapid changes of the link conditions. We resolve this problem by two solutions. In the first place, we have developed a responsive MAC adaptation method using SNR and packet-loss statistics. The second solution depends on the first one, it allows the MAC to communicate information about changes of link conditions to the video codec. The codec then utilizes this information to adapt its data rate accordingly. The two solutions work together in harmony to minimize the effects of the unstable and unreliable radio connection, greatly improving the quality of the decoded video.

In this chapter we detail how the prediction of the real data throughput (needed by the video codec) is performed by the MAC layer. This includes the (more realistic) case when the medium is shared with other stations.

4.1 Real throughput in 802.11

4.1.1 Introduction

Most often, when the Link Layer has to provide information to upper layers, this information includes available and/or predicted throughput. The throughput that the applications see is dependent on many factors. In 802.11 the most important factors are the selected Radio Rate, the sharing of the medium, the length of the packets, the link conditions, and the usage of the RTS/CTS scheme.

Our link layer too provides throughput information to the application layer. From practical experience we derived a number of requirements, which the throughput prediction model that is employed at the link layer should meet:

1. Low computational complexity: Since all the throughput calculations are performed in real-time either on the firmware level or on the card driver level, the complexity and the number of calculations should be kept as low as possible.

2. Low memory utilization: This requirement has the same grounds as the previous one. It translates into avoiding the use of big histories of data (such as statistics history for example).

3. Reasonable accuracy: At the same time, the model should provide throughput values that are accurate enough for the target conditions. From a practical point of view, we require that the relative prediction error to be equal to or less than 20%.

There are a lot of publications devoted to throughput analysis in IEEE 802.11 (for example [Bia00; Qia02; Che04a]). However, all of them are oriented towards simulation, i.e. they stress on the accuracy of the model, which is paid by an extensive model complexity. As such, these models do not meet our requirements. Therefore, we have used the 802.11 standard [IEE11] to derive a model ourselves. To simplify the model, and consequently, its computational complexity, we make the following assumptions:

1. Distributed Coordination Function (DCF) is used for all medium access.

2. There is no medium sharing - i.e. no other station is using the same channel at the same time.

3. The link quality is good.

4. The transmission queue of the radio is always non-empty, i.e. after each frame sent, another frame is immediately attempted for sending.

(Note that assumptions 2 and 3 imply that there are no retransmissions.)

While with these assumptions we derive the maximum throughput ever available to the application layer (i.e. the throughput upper bound), we can easily compensate the introduced inaccuracy by applying correcting coefficients. From a practical point of view this method is preferred to the case in which the model is more accurate, but imposes a significant computational burden.

4.1.2 Model derivation

To calculate the throughput, we need to know the time T_{PKT} that it takes a packet to be transmitted, the total amount of data L per packet, and the rate setting D selected for transmission of this packet:

$$Throughput(D, L) = \frac{L}{T_{PKT}(D, L)} \tag{4.1}$$

In 802.11 a packet is encapsulated in a MSDU (MAC service data unit), which is the basic data unit transmitted over the air. So, the time to transmit a packet is the time to transmit a MSDU, or $T_{PKT} = T_{MSDU}$.

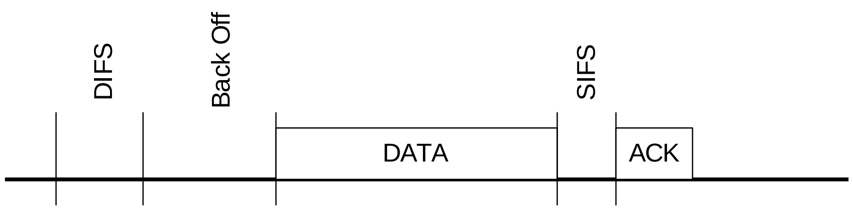


Figure 4.1: *Single data frame transmission in 802.11.*

A typical frame transmission under DCF is shown in Figure 4.1. The time to transmit a frame is:

$$T_{MSDU} = T_{DIFS} + T_{BO} + T_{DATA} + T_{SIFS} + T_{ACK}$$

where T_{DIFS} is the distributed (coordination function) interframe space (DIFS) time, T_{BO} is the backoff time, T_{DATA} is the time the actual data is transmitted for, T_{SIFS} is the short interframe space (SIFS) time, and T_{ACK} is the time for the acknowledgement. Here, we make the assumption that no fragmentation is performed - that is, the MSDU is not divided in MPDUs (MAC Protocol Data Units).

T_{DIFS} and T_{BO} are defined by the 802.11 standard as:

$$T_{DIFS} = T_{SIFS} + 2T_{SlotTime}$$

$$T_{BO} = Random() \times T_{SlotTime}$$

Since T_{BO} is a random number with uniform distribution in the interval $[0, CW_{min}]$, we will use its average, assuming that collisions do not occur:

$$T_{BOav} = \frac{CW_{min} \times T_{SlotTime}}{2}$$

The calculation of T_{DATA} depends on the technology (802.11b, 802.11g, or 802.11a). Therefore, we designate T_{DATA} for the different technologies as T_{DATA_11a} and T_{DATA_11b} accordingly (we do not consider the calculation for 802.11g as it is similar to the one for 802.11a). All the times calculated will be given in μs.

For T_{DATA_11a} we have (see [IEE11a], Section 17.4.3, Formula 30):

$$T_{DATA\_11a} = T_{PREAMBLE} + T_{SIGNAL} + \frac{16 + 8L + 6}{D} + \frac{T_{SYM}}{2} + T_{err} \tag{4.2}$$

where the times $T_{PREAMBLE}$, T_{SIGNAL}, and T_{SYM} are defined by the standard, L is the length of the packet in bytes, and D is the Rate Setting in Mbits/s. T_{err} is an additional term that we introduce that compensates for the error caused by the effect of rounding to the next OFDM symbol. This error could be in the range of $\pm\, 2\,\mu s$. To account for the worst-case scenario, we use $T_{err} = 2\,\mu s$.

	802.11a	802.11b
T_{DIFS}	34 μs	50 μs
T_{SIFS}	16 μs	10 μs
$T_{SlotTime}$	9 μs	20 μs
T_{BO}	67.5 μs	310 μs
CW_{min}	15	31
$T_{PREAMBLE}$	16	144 μs/72 μs
T_{SIGNAL}	4	-
$T_{PLCPHeader}$	-	48 μs/24 μs

Table 4.1: *Some parameters in 802.11.*

For T_{DATA_11b} we have (see [IEE11b], Section 18.3.4):

$$T_{DATA\_11b} = T_{PREAMBLE} + T_{PLCPHeader} + \lceil \frac{8(L + PBCC)}{D} \rceil \tag{4.3}$$

where $T_{PREAMBLE}$ and $T_{PLCPHeader}$ are again defined by the standard. Here also the values of those two terms depend on what kind of PLCP preamble and header are selected. The 802.11b standard provides the option of using short PLCP preambles and headers instead of the normal (long) PLCP preambles and headers (see [IEE11b], Sections 18.2.1 through 18.2.3). This option allows for achieving higher goodput at the price of possible incompatibility with legacy non-short-preamble capable equipment. The term $PBCC$ is 1 if the optional PBCC (packet binary convolutional code) modulation scheme is used, 0 if not. Since in practice the 802.11 manufacturers did not agree to implement the PBCC option, we can safely assume that $PBCC = 0$.

Using the values of the parameters given in Table 4.1, as defined in the 802.11 standard [IEE11; IEE11a; IEE11b], we get for 802.11a

$$T_{DATA\_11a} = 22 + \frac{22 + 8L}{D}$$

and for 802.11b

$$T_{DATA\_11bLP} = 192 + \frac{8L}{D}$$

for the long preamble case and

$$T_{DATA\_11bSP} = 96 + \frac{8L}{D}$$

for the short preamble case.

Finally, we can calculate the throughput for 802.11a, in Mbits/s:

$$Throughput\_11a = \frac{8L}{34 + 67.5 + 16 + 22 + (22 + 8L)/D + 22 + 134/D}$$

$$Throughput\_11a = \frac{8L}{\frac{8}{D}L + \frac{156}{D} + 161.5} \tag{4.4}$$

By similar calculations for 802.11b, we get, in Mbits/s:

$$Throughput\_11bLP = \frac{8L}{\frac{8}{D}L + \frac{112}{D} + 754} \tag{4.5}$$

for the long preamble case and

$$Throughput\_11bSP = \frac{8L}{\frac{8}{D}L + \frac{112}{D} + 562} \tag{4.6}$$

for the short preamble case.

We conducted a number of experiments to determine the accuracy of our model. We used netperf to measure the maximum throughput between two stations, achievable at each rate setting. The stations were equipped with 802.11a and 802.11b cards, and were situated close to each other, so that the link quality is always good. There was no activity from other stations present. The results from the experiments are shown in Figure 4.2 for 802.11a, and in Figure 4.3

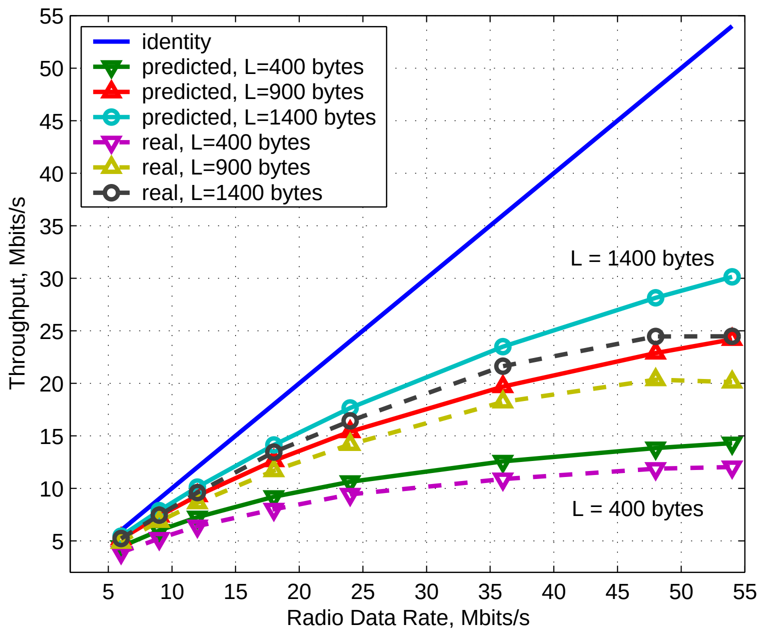


Figure 4.2: *Predicted versus real throughput for 802.11a.*

for 802.11b. In both figures the gap between radio data rate and the achieved throughput is easily observed. The gap, which is due to the MAC data overhead, increases when increasing data rate, and also when decreasing the packet size. Note that there is a possible tradeoff here. While smaller packets result in lower

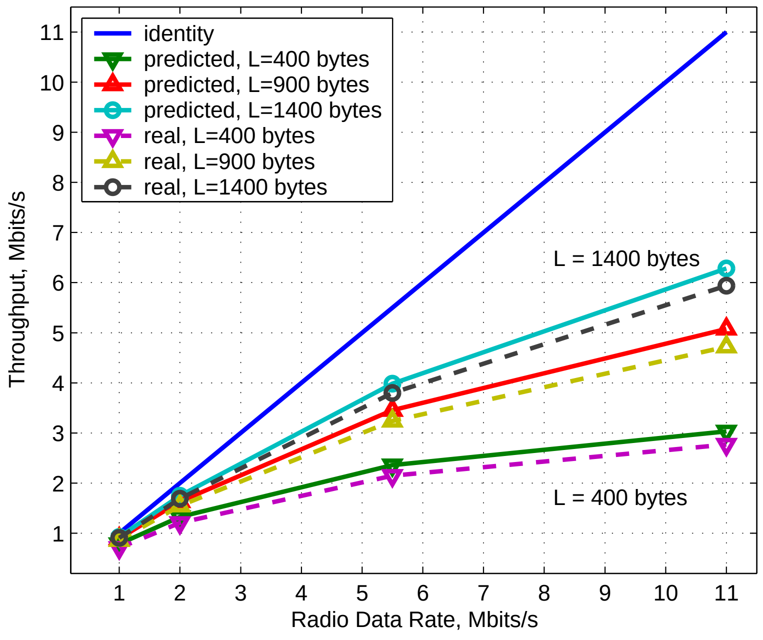


Figure 4.3: *Predicted versus real throughput for 802.11b.*

throughput in ideal link conditions, the possibility of a packet being successfully transmitted over a bad link is inversely proportional to its size. Thus, in reality there could be situations where smaller packets would actually result in the same, or even bigger throughput than achieved by using larger packets.

The maximum error for 802.11a prediction does not exceed 19%, with an average error of less than 10%. We observed a strange dip in the throughput for large packets, at the 54 Mbits/s rate setting. Our investigation pointed out that this is most probably due to a chipset/firmware issue, since we have made tests with different drivers and OSs, which all yielded similar results. Unfortunately, because of the unavailability of other 802.11a equipment, we could not make comparisons with other chipsets. For 802.11b the results are even better, with 10% maximum error and 7% average error. Overall, we have met the requirements stated in Section 4.1.1, so we are going to use this model in our practical implementations.

4.2 Non-shared medium case*

The quality of real-time video streaming over wireless links is significantly reduced due to packet losses and delays caused by rapid changes of the link conditions.

*This section has been published as "Fast 802.11 link adaptation for real-time video streaming by cross-layer signaling" by I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips in Proceedings of the *IEEE Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005.

We resolve this problem by two solutions. In the first place, we have developed a responsive MAC adaptation method using SNR and packet loss statistics. The second solution depends on the first one, it allows the MAC to communicate information about changes of link conditions to the video codec. Together, these solutions will prevent packet losses and skipped frames, both of which have an adverse effect on the visual quality of the decoded video. In this section we present the results of an streaming video experiment using an 802.11a link between a fixed an a mobile station. We show that with the cross-layer signaling between the MAC-layer and the video coder, an increase of 4dB visual quality is achievable.

Keywords: video streaming, rate control, MAC layer, SNR, link adaptation, cross-layer interaction

4.2.1 Introduction

With the boom of mobile communications, IEEE 802.11 seems to become a corner-stone in wireless Internet access. Also, end-users are increasingly making use of multimedia streaming applications, like voice-over-IP and video conferencing. As a consequence, there is a demand that multimedia streaming is well supported over wireless connections. Such connections usually provide far less quality and stability than their wired counterparts. The shortcomings of wireless connections are due to two reasons: First, the radio channel is a shared medium and the frequency spectrum is limited, so both bandwidth (data rate) and transmitted power (error rate) are constrained. Secondly, in a mobile environment, the radio channel conditions change frequently because of the varying distance between stations, Rayleigh fading, and interference. As a result of these inherent problems of the wireless connections, packet delay, jitter and losses are significantly higher than in wired connections. Since multimedia applications are extremely sensitive to such packet delivery problems, mobile users experience reduced performance.

To handle the effects due to changes in the channel conditions, typically so called link adaptation (LA) is being employed at the link (MAC) level. Basically, LA is a process of automatically adjusting a number of radio/MAC parameters, so that optimal quality of packet transmission is achieved. Here optimal quality means minimizing packet loss rate (PLR) and packet delay, while keeping the data throughput as high as possible. Too often though, LA is oriented towards download-type of applications. So the emphasis is on maximizing the throughput, while minimizing the delay and PLR is a task of secondary importance. This hinders the performance in the case of multimedia streaming.

At the application layer, the video encoder can also adapt to the link quality by changing the compression degree for example, and thus modifying the data rate. This adaptation requires that the video encoder is able to sense the link quality, for example, by getting a feedback information from the decoder side.

However, such a scheme is ineffective when the round-trip delays are too long. Using this approach also introduces additional overhead.

Our solution is to combine the above methods. The video encoder will adapt based on information provided by the LA entity that is at the radio level. The information consists of the current link status report, plus some additional forecast about what the link conditions are going to be in the next couple of tens of milliseconds. In [Taa02; Taa03] we already have done experiments with an adaptive video coder and MAC-layer based on QoS negotiations [Dij00], but without tight coupling of the rate-control loops.

The remainder of the section is organized as follows. The adaptive video coder is discussed in the next section. Section 4.2.3 gives a description of the basics of the link adaptation, along with our hybrid rate-control solution. The inter-layer communication is discussed in Section 4.2.4. The experimental results are discussed in Section 4.2.5. The conclusions are presented in Section 4.2.6.

4.2.2 Adaptive Video coding

We use a video coder that is able to adapt to a changing channel. The video codec we used, is a H.263 codec that is modified to support interaction with other protocol layers such as in this case the MAC layer. Our version of the H.263 encoder supports a (video) rate-control algorithm (VRCA) that tries to achieve a certain rate, by adjusting the quantization step size. The quantization step size is the main parameter that controls the compression of the video. Although this VRCA is designed for constant bit rate (CBR) encoding, it can also be used to dynamically change the bit rate that is produced by the encoder. Since the resulting bit rate depends on the selected quantization step size but also on the statistics of the picture itself, we cannot — unfortunately — set the bit rate beforehand, and then expect that this bit rate will be exactly achieved by the encoder. The VRCA, is a simple feedback control loop that consists of setting an initial quantization step size, encoding part of the picture, measure the resulting intermediate bitrate, increase or decrease the step size accordingly and then continue with the next part of the picture. In total there are nine parts of a picture frame for which the quantization step size can be adjusted, which generally is enough to able to achieve a certain preset rate. This algorithm works fine in the CBR case and with slowly varying picture statistics. In principle we can change the target bit rate for each individual frame, meaning that we have a maximum delay of 100 ms to respond to changes in the channel. In fact, we can even change the target bit rate for each part *within* a frame, so we can even faster adjust to the channel conditions. In our experiment, we will constantly adapt the target video encoding rate for the VRCA according to information from the MAC rate-control algorithm.

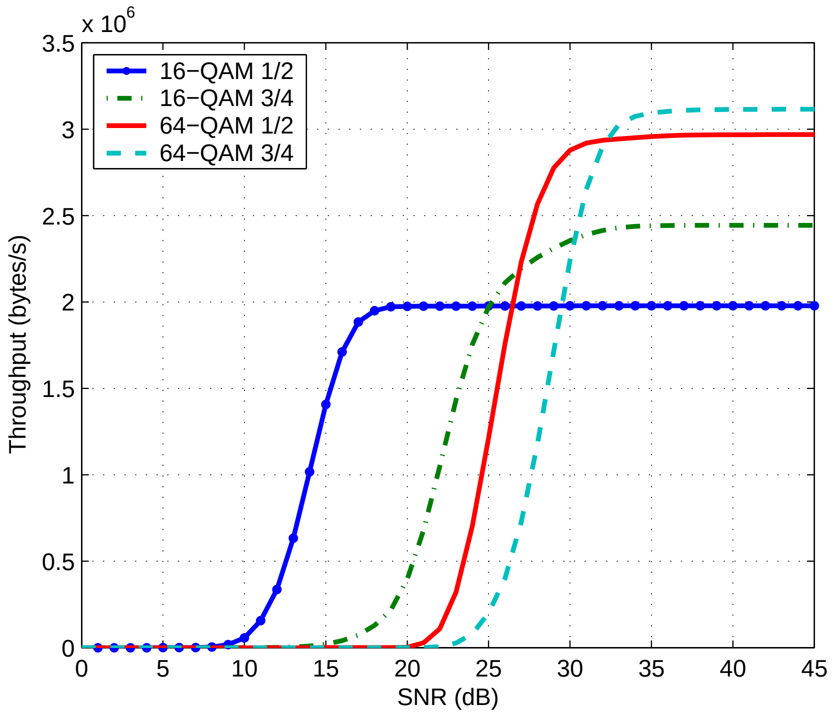


Figure 4.4: *Throughput vs. SNR for some 802.11a modulation schemes.*

4.2.3 MAC Link adaptation

The IEEE 802.11 standard defines several MAC-level parameters (settings) that are available for tuning at the side of the Wireless Network Interface Card (NIC). The most important parameter available for adjustment is the transmit rate. In 802.11a, for example, it can be set to 6, 9, 12, 18, 24, 36, 48 and 54 Mb/s. Each rate corresponds to a different modulation scheme with its own trade-off between data throughput and distance between the stations. This can be seen in Figure 4.4 – for clarity only the last four modulation schemes are shown. It shows the performance in terms of the throughput for each modulation scheme available in IEEE 802.11a versus the signal-to-noise ratio (SNR). Note that distance is related to SNR as $SNR \sim \frac{1}{dist^{\alpha}}$. More complex modulation schemes like 64-QAM 3/4 offer a larger throughput, but also have increased sensitivity to channel noise, and thus provide a shorter operating range. Usually, one wants to extend the operating range as much as possible and, at the same time, to maximize the throughput. This can be done by proper (automatic) selection of the rate (modulation scheme) that gives the maximum throughput for certain conditions, for example, by selecting 64-QAM 1/2 at an SNR of 30 dB (Figure 4.4).

In order to decide which rate is optimal at each specific moment, a control algorithm needs information about the current link conditions, or so-called channel state information (CSI). Since it is difficult to get CSI directly, most MAC rate-control (MRC) algorithms use some form of statistics-based feedback, for example, user-level throughput. The main disadvantage of this indirect feedback is that it is inherently slow, causing communication drop-outs when the link conditions degrade rapidly (e.g., when the user moves fast). The short-term drop-outs are normally handled by frame retransmissions. This is acceptable for download applications whose main requirement is gross data throughput. The retransmissions, however, lead to a significant increase in (average) packet delay, and the variations in the number of retransmissions cause an increase in the jitter of the packet delay. Streaming applications are very sensitive to long packet delays and high jitter, and less sensitive to the overall throughput of the link (provided of course that this throughput is larger than the minimum throughput that the application requires). Consequently, streaming applications perform poorly under standard automatic MRC. In [Har03], for example, it is reported that switching from automatic MRC to a manually selected fixed rate extends the maximum distance between stations up to 40% for the flawless display of a video stream over an 802.11a wireless connection.

A logical way to cope with the slow accommodation characteristics of statistics-based feedback methods is to look for methods that use faster feedback, i.e., feedback that quickly provides up-to-date information about the channel status. Such a feedback — the SNR — has been theoretically discussed in previous works, but so far, to our knowledge, it has been used in only one practical implementation [Har04]. We use this LA control algorithm in our 802.11 radio to enhance multimedia performance, and also to provide feedback information about the channel conditions that the video encoder requires.

4.2.4 Layer interaction

In our realtime video transmission scenario, the video rate control algorithm depends on information from the MAC rate control algorithm. By coupling the rate control algorithms of the MAC and the video coder in this manner we expect to efficiently use the available transmission rate to maximize the picture quality.

In the envisioned scenario, we typically encode a video sequence at ten frames per second. Each video frame is split up in nine slices (a horizontal group of blocks). For each slice, the quantization step size can be changed by the VRCA. The MAC-layer rate control algorithm can inform the VRCA about the current transmission rate on a packet basis, such that the VRCA can adapt the target encoding rate for the current frame (Figure 4.5). This may have several drawbacks. 1) If the transmission rate fluctuates, this results in a picture with fluctuating quality. 2) The VRCA may not be able to follow fast changes in the transmission rate. Sudden large drops or peaks in the expected transmission rate, may then result in a video encoding rate that not matches the transmission rate.

Both observations led us to conclude that we need a longer term prediction

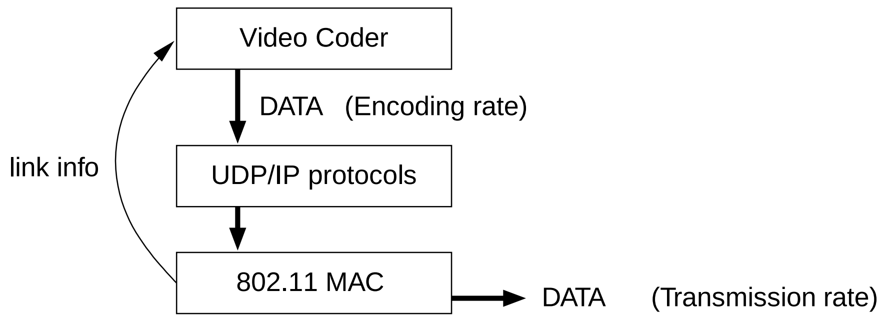


Figure 4.5: *Video layer, UDP/IP layer and the MAC layer. The MAC gives link information to the video layer at request of the video layer.*

of the transmission rate, besides the short-term or next-packet transmission rate. The validity of this prediction should then be in the order of one video frame or 100ms. On the one hand, we can use this long term prediction to more accurately initialize and adapt the quantization step size. On the other hand, this long term prediction is not always accurate enough. The inaccuracy is no problem since we can still adapt at a later stage, using the next-packet transmission rate. In stationary circumstances the long-term prediction will be quite accurate. On non-stationary circumstances, however, a large over-estimate of the transmission rate may result in a encoding rate that is too large, causing buffer overflows and large delays. Large under-estimates are not so bad, since the encoding rate is then lower than the transmission rate and all data will arrive on time, only resulting in a little lower picture quality than necessary. This means that we rather have an underestimate than an overestimate of the future rate. We therefore use a scheme as shown in Figure 4.6, where the MAC rate-control delivers two expected transmission rates. 1) The transmission rate for the next packet (accurate). 2) A conservative expectation of the average transmission rate over a time span of 50-100ms. The VCRA then uses these two values for fast adaptation of the video encoding process.

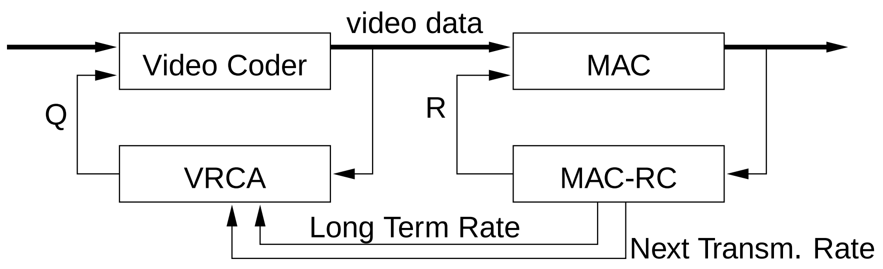


Figure 4.6: *Simplified overview of the coupled Rate control scheme. The rate control mechanisms of the video encoder and the MAC are coupled by the flow of information about the MAC transmission rates. For simplicity the UDP/IP layers are discarded in this picture.*

On the MAC side the predictor uses history of SSIA (Signal Strength Indication of Acknowledgements) measurements to maintain a short-term (in order of few tens of milliseconds) SSIA average. Over this average future SSIA are predicted. Our experiments show that the best results are obtained by first or zero-th order predictor. This conclusion agrees with the theory as well, the latter stating that channel state change is a memoryless process.

The SSIA predictions are matched against the SSIA thresholds in the SSIA-rate lookup table in the radio rate-control algorithm [Har04], and so the rate predictions are determined.

Our experiments show, that for the dynamic case (the user moves intensively), the accuracy of the rate prediction for 100ms ahead is better than 80%. And the probability that the predicted rate will be within the real rate ± 1 is larger than 90%.

4.2.5 Experiments

The experiments were carried out by streaming a video file between two laptops, both running Linux. The 802.11a cards used are based on the Atheros AR5000 chipset, and the card driver uses the advanced hybrid rate control algorithm, as described in [Har04]. One of the laptops had a fixed position and the other one is following a predetermined track. The track consists of three parts - "lead-in", which is reaching from the room to a specific start position in the hallway, and waiting until certain time elapses (10s). Then follows moving with the laptop three times up and down the hallway (60s). Finally we go back ("lead-out") again into the room, where the fixed laptop lies (20s).

To evaluate the performance of the coupled rate-control systems, we have evaluated three cases. Each experiment took 90 seconds. We managed to quite precisely time the experiments and to obtain a reasonable repeatability of the results. Later in this section we will compare the quality of the received videos using the Peek-signal-to-noise-ratio (PSNR), a commonly used metric in video compression.

1. **Coupled** : Coupled rate control. The rate control loops of the MAC-layer and the Video coder are coupled by letting the video coder poll frequently for the current and predicted rate.

2. **De-Coupled-MAX** : De-coupled rate control. Since the video now has no indication of the actual rate, we set the target-rate for the VRCA to the *maximum* rate as was obtained from the VRCA in the Coupled case.

3. **De-Coupled-AVG** : De-coupled rate control. Here we set the target-rate for the VRCA to the *average* rate as was obtained from the VRCA in the Coupled case.

In the De-Coupled-MAX case, we expect to obtain a higher quality than in the De-Coupled-AVG case. However, when the actual rate is lower than the preset target-rate, the encoder is not able to stream all data in time. The encoder will then have to skip frames[1], which will result in a lower PSNR and a lower visual quality at the receiving side. In the De-Coupled-AVG we expect an overall lower quality, since it is coded at an overall lower rate. On the other hand, we expect less skipped frames to occur.

Case	# Skipped frames	# Lost Packets	PSNR	PSNR Middle
Coupled	50	13	40.5	39.3
De-Coupled-MAX	193	19	37.8	35.4
De-Coupled-AVG	216	23	36.3	33.1

Table 4.2: *Results for the three cases. The column "PSNR" shows the average PSNR over the whole sequence. The column "PSNR Middle" shows the average PSNR during the "Changing conditions period" between 10 and 70 seconds.*

In Figure 4.7 the quality (PSNR) is shown for the whole experiment (90s). In the left part ($0-10$s) the channel conditions are excellent, hence the high quality in all cases. The middle part is best described as having conditions changing from good to bad a few times. The right part has good conditions again. In Figure 4.8 the received-signal strength is shown for the same time span. The pattern clearly corresponds to the followed track. Note that the strength already drops quite strongly in the first (0-10s) part. However, down to $30-35$dB the highest possible rate (at the radio) provides excellent performance.

As we can see in Figure 4.7, the Coupled case has a higher PSNR in almost all cases. Table 4.2 summarizes the number of skipped frames (by the encoder), the number of lost packets, the average PSNR and the average PSNR in the period between 10-70s. Although the number of lost packets are not the same in the three cases, the differences do not justify the differences in PSNR. In the decoupled cases, the total number of skipped frames is much higher, as we expected. Looking at the average PSNR in the $10-70$s period, we conclude that the quality can be dramatically improved by informing the video codec of the actual present rate and a prediction for the near future.

4.2.6 Conclusions

In this section we have presented the results of a streaming video experiment over a wireless 802.11a connection between two laptops, one of which was moving. The

[1]The encoder has two options in this case. 1) Skip a frame such that the next frame can arrive in time, effectively lowering the rate 2) Just keep on encoding at the same rate, which will result in buffer overflows and lost packets. Both options will result in degraded visual quality. We have chosen the first option

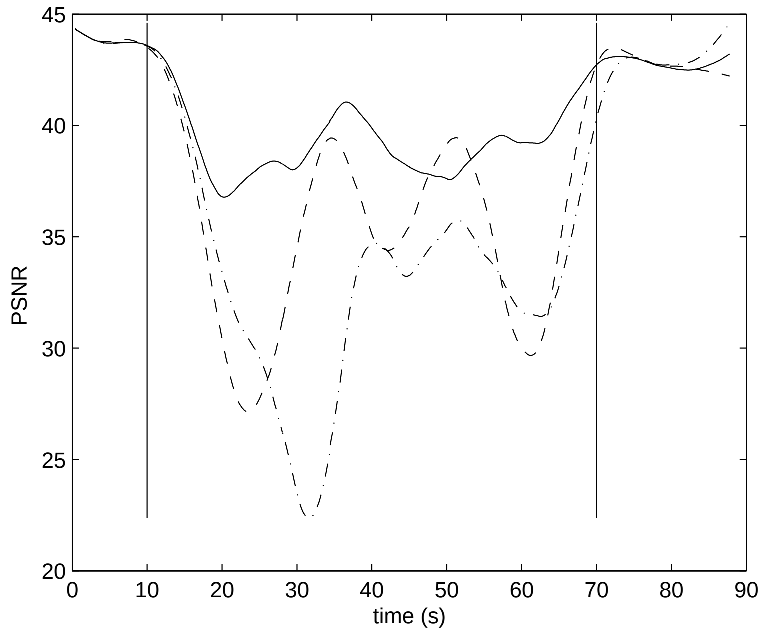


Figure 4.7: *PSNR: the solid line shows how the quality (PSNR) changes during the experiment for the Coupled. The dashed line shows the results for a decoupled rate control using the maximum rate (De-Coupled-MAX). The dash-dotted line shows for the decoupled rate control using the average rate (De-Coupled-AVG). We have applied an averaging sliding window to the data for better visibility.*

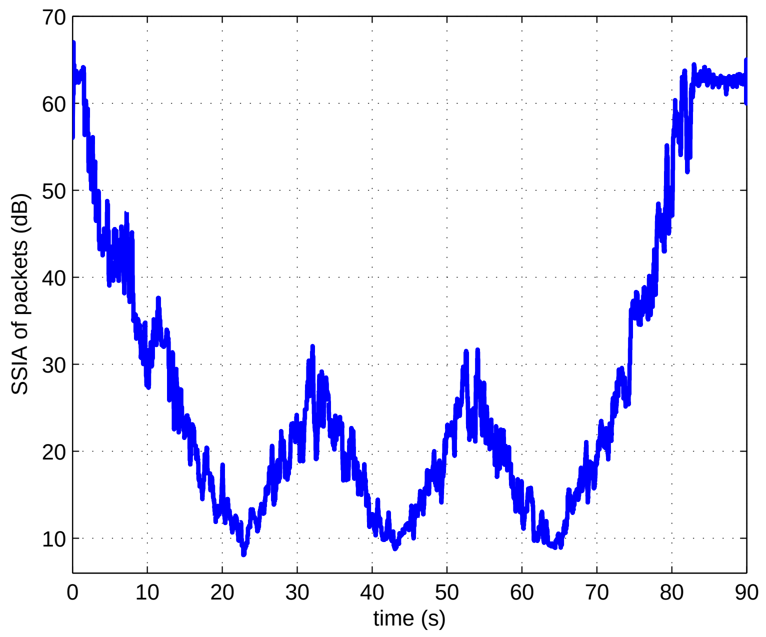


Figure 4.8: *Measured SSIA during experiments.*

changing channel conditions make that the video cannot always be transmitted at the full rate, without packet losses. To handle the changing conditions and packet losses we implemented two solutions. 1) responsive MAC-adaptation using (radio-)SNR and packet loss statistics. 2) cross-layer signalling of changing channel conditions between the MAC-layer and the video coder. The first solution tries to prevent packet-losses and transmit buffers running full, by lowering the transmission rate when high packet-losses are expected. The second informs the video codec of these changes such that the transmission rate is matched with the generated video data rate. As a result, our video streaming experiments suffered from a very small number of packet losses and also a small number of skipped frames. Both effects resulted in a high visual quality as opposed to the case where there is no cross-layer signalling. In changing channel conditions, an increase of 4dB PSNR or more is a realistic figure.

4.3 Shared medium case[*]

Changes in the quality of wireless links impose great demands on video codecs and underlying network layers when seamless video-streaming is to be achieved. Moreover, it is not enough that only the video codec or only the radio adapts to these changes; the efforts should be applied in both layers, and – if possible – synchronized. So, on the one hand, a responsive link adaptation method should be employed in radio. And, on the other hand, the video codec should be able to follow the changes in the maximum throughput due to wireless link performance variations. In this section we present the results of video-streaming over 802.11a link in the presence of background traffic, generated by other stations sharing the same medium. We show that great improvements in the quality of the video can be achieved by cross-layer signaling between the link layer and the video coder. However, we show that this is only realizable if correct estimation can be made of the throughput decline due to the medium sharing.

Keywords: video streaming, medium sharing, rate control, MAC layer, SNR, link adaptation, cross-layer interaction, 802.11

4.3.1 Introduction

Having a wireless last hop in the Internet is something that becomes more and more popular today, and so is using the Internet for audio and video streaming. The combination makes that the demanding world of real-time multimedia (which

[*]This section has been published as "Link Adaptation and Cross-Layer Signaling for Wireless Video-Streaming in a Shared Medium" by I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips in Proceedings of the *IEEE International Conference on Wireless Networks, Communication and Mobile Computing (WirelessCom)*, Maui, Hawaii, USA, June 2005.

does not tolerate things like drop-outs for example) meets the quite imperfect – and capricious – dark universe of radio links. A lot of effort is required to team these worlds together, so that the stringent packet delay, jitter, and loss requirements of multimedia applications can be met by the unstable and unreliable radio link.

Different approaches exist to keep the marriage of multimedia and wireless happy. First, the so called link adaptation (LA) is typically being applied at the link (MAC) level [PP03; Kam97; Veg02; Bal99]. Roughly speaking, LA is the process of automatic selection of radio/MAC parameters, so that optimal quality of packet transmission is achieved. By optimal quality we mean minimizing packet loss and packet delay, while keeping the data throughput as high as possible.

Second, the video encoder can also adapt to the link quality, for example, by changing the compression degree, and thus modifying the (video) data rate [Bol98; Ort95]. This adaptation requires that the video encoder is able to sense the link quality, for example, by getting feedback information from the decoder side. However, such a scheme is ineffective when the round-trip delays are long. Using this approach also introduces additional overhead. A better approach, described in [Har05b], for the video encoder is to base the adaptation on information provided by the link adaptation entity at the radio level.

It is important to note here that control schemes in the lower layer (i.e. at the MAC level) have precedence in terms of achieved improvement in performance, to control schemes in the upper layer (the video codec). In other words, and to put it simply, there is no way that you can compensate for packets already lost at the link level.

Another significant observation is that the performance gain of the whole system can be nullified in situations where the information provided by the lower layer to the upper layer is incorrect. In our case failing to consider the sharing of the medium in the throughput estimation of the wireless channel can be fatal, as we show in this section.

We present results of real experiments that show how much performance can be gained by using an ideal throughput estimator. Although we emulate the estimator (by knowing beforehand what would be the available throughput while sharing the medium), we present an idea of how such an estimator, having sufficient prediction accuracy, can be built in practice. Building and testing the complete estimator, however, constitutes future work.

The remainder of the section is organized as follows. In the next section, a discussion is presented about the degree of control at codec and MAC side, and their inter-operation, depending on the harshness of the environment and the posed constraints. Section 4.3.3 gives a description of the adaptive video coder and the basics of link adaptation. The cross-layer communication and the medium sharing prediction are discussed in Section 4.3.4. The experimental results are discussed in Section 4.3.5. The conclusions and future plans are presented in Section 4.3.6.

4.3.2 Control complexity as function of constraints severity

When designing a communication system, it is important to know what are the inherent problems in the communication link and their severity. Then, given the constraints that the application imposes, such as the performance quality variations that it could handle, an appropriate control mechanism is applied. On the one hand this mechanism should assure that the quality variation constraints are kept, and on the other, it is desirable to keep the complexity of this mechanism as low as possible. Keeping the complexity of the control low minimizes the design efforts, and, in most of the cases, also guarantees maximum robustness of the whole system, compared to the case with more-complex control.

In Figure 4.9 we show 3 cases of different control complexity, which are the basis of our experiments. Case 1 depicts the situation when there is a need for LA, in which situation some kind of channel state prediction is employed. In case 2, the channel state information (CSI) is also supplied to the video codec via simple radio throughput to real throughput function. This information is used by the codec to adapt the video rate accordingly, so that the video data does not choke the radio link. The most complex is case 3 where the real link throughput is calculated from the CSI by a medium sharing predictor or estimator. As shown below, this is very important in the case there are other 802.11 users in the same BSS, for example.

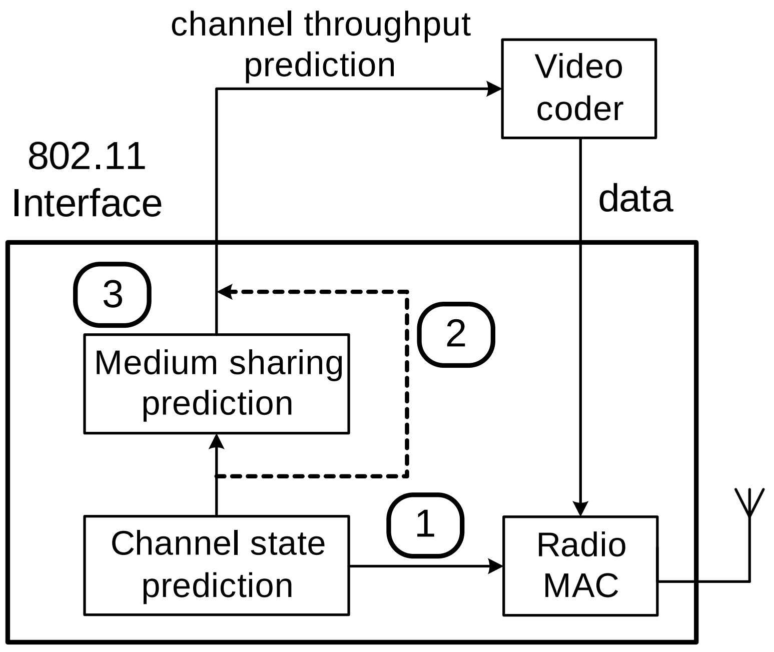


Figure 4.9: *Cases of prediction. 1) – Channel state prediction. Used for Link Adaptation. 2) – Channel throughput prediction without considering medium sharing. 3) – Channel throughput prediction considering medium sharing.*

In Table 4.3 we present the minimal degree of control complexity, depending on the video requirements and the environment conditions. Let us look at the upper part first (no medium sharing). In the left half we have the cases when the channel is static, i.e. there is no movement or any change in the link quality. Then we do not need channel state prediction, and we do not have to change any parameter in the radio. A typical example of such links are satellite links. However, in our case, the channel is dynamic, so we fall into the right half of the table. There we definitely need channel state prediction to be able to adapt to the channel quality variations. Failing to do so will have catastrophic consequences, no matter how smart the network layers above are, just because broken radio link means no packets arrive at all.

Medium	Static channel		Dynamic channel (movement)	
	Low quality video	High quality video	Low quality video	High quality video
No sharing	-	-	1	1+2
Sharing	(3)	3	1+3	1+3

Table 4.3: *Minimal system configuration (see Figure 4.9) for different requirements, necessary for providing seamless video-streaming.*

Moreover, in the case of high quality video requirements, we also need the codec to have some information about the available throughput. The rationale behind this is discussed in [Har05b].

The picture changes completely, if the wireless channel is shared. In Figure 4.10 we compare the throughput available to a typical video-streaming application, with, and without background download (TCP) traffic present. For completeness we also present the predicted throughput value, given by the prediction model presented in Section 4.3.4. As it is shown in the Figure, the throughput can drop significantly, even with only one more user who is downloading. Therefore, we need medium sharing prediction to be employed as well, for all cases.

The data in Figure 4.10 (except the predicted values) is obtained by real-time experiments, where the video-streaming and the download traffics are emulated by using NetPerf. The UDP packet size is chosen to match the average packet size generated by our video encoder in the experiments in Section 4.3.5.

4.3.3 MAC Link Adaptation and Adaptive Video Coding

In the IEEE 802.11 standard [IEE11] several MAC-level parameters are defined, which are available for tuning at the side of the Wireless Network Interface Card (NIC). From those parameters, the most important one available for tuning is the transmit rate. In 802.11a [IEE11a], for example, it can be set to 6, 9, 12, 18, 24, 36, 48 and 54 Mbits/s. Each rate setting offers a different trade-off between data

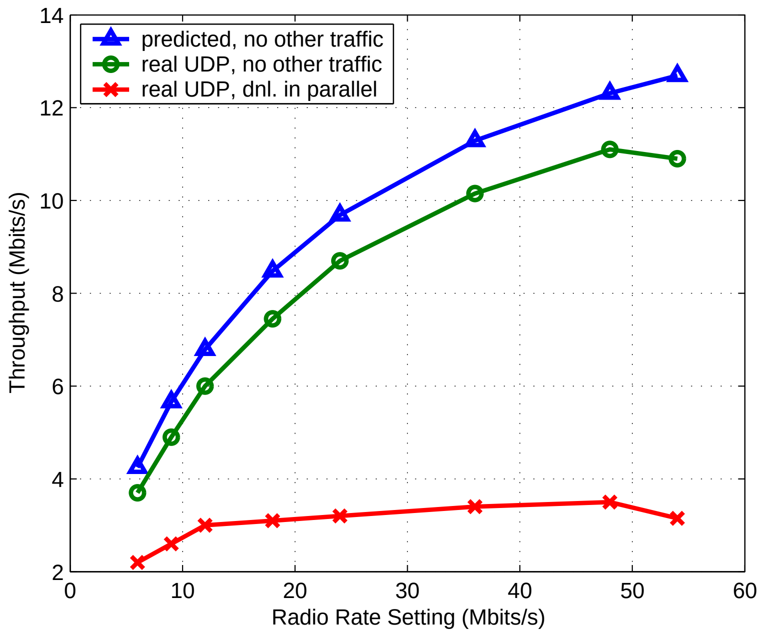


Figure 4.10: *Available throughput for a video-streaming application, with, and without background download (TCP) traffic, compared with predicted value as well. The used UDP packet size is* 341 *bytes.*

throughput and achievable distance between stations. Bigger rate setting offer a larger throughput, but also provide a shorter operating range. Usually, one wants to extend the operating range as much as possible and, at the same time, to maximize the throughput. This can be done by proper (automatic) selection of the rate.

In order to decide which rate is optimal at each specific moment, a control algorithm needs information about the current link conditions, or so-called channel state information (CSI). Since it is difficult to get CSI directly, most MAC rate-control algorithms use some form of statistics-based feedback, for example, user-level throughput. The main disadvantage of this indirect feedback is that it is inherently slow, causing communication drop-outs when the link conditions degrade rapidly (e.g., when the user moves fast). Streaming applications are very sensitive to these drop-outs. Consequently, streaming applications perform poorly under standard automatic rate-control.

To achieve fast accommodation characteristics, we use the SNR-based hybrid rate controller, which we describe in a previous work [Har05a]. We also use the predictor that we developed in [Har05b] to provide feedback information about the channel conditions to the video encoder.

We use a H.263 video codec that is modified to support interaction with other protocol layers. Our version of the H.263 encoder supports a (video) rate-control algorithm (VRCA) that tries to achieve a certain rate, by adjusting the quantization step size. The quantization step size is the main parameter that controls the compression of the video. This VRCA was designed for constant bit rate (CBR) encoding, but it can also be used to dynamically change the bit rate that is produced by the encoder. Since the resulting bit rate depends on the selected quantization step size but also on the statistics of the picture itself, we cannot set the bit rate beforehand, and then expect that this bit rate will be exactly achieved by the encoder. The VRCA, is a simple feedback control loop that consists of setting an initial quantization step size, encoding part of the picture, measuring the resulting intermediate bitrate, changing the step size accordingly and then continuing with the next part of the picture. In total there are nine parts of a picture frame for which the quantization step size can be adjusted, which generally is enough to able to achieve a certain preset rate. With this algorithm, we can change the target bit rate for each individual frame, meaning that we have a maximum delay of 40 ms to respond to changes in the channel. In our experiments, we adapt the target video encoding rate for the VRCA, based on the feedback information provided by the 802.11 link layer.

4.3.4 Cross-Layer Signaling and Medium Sharing Prediction

As discussed in the previous section, the VCRA uses throughput prediction information, provided by the 802.11 MAC. This information is generated in the following way.

On the MAC side the predictor uses history of SSIA (Signal Strength Indication of Acknowledgements) measurements to maintain a short-term (in order of few tens of milliseconds) SSIA average. Over this average future SSIA are predicted. Our experiments show that the best results are obtained by first or zero-th order predictor. This conclusion agrees with the theory as well, the latter stating that channel state change is a memoryless process.

The SSIA predictions are matched against the SSIA thresholds in the SSIA-rate lookup table in the radio rate-control algorithm [Har05a], and so the rate predictions are determined.

Then, for the case with no sharing of the medium we use a simple model, to convert from radio rate setting to available user-space data throughput. Our model is derived from the IEEE 802.11 standards (see [IEE11; IEE11a; IEE11b]), and is in the form of:

$$T = \frac{8RL}{8L + bR + c}$$

where T is the throughput in Mbits/s, L is the length of a packet in bytes, R is the data rate setting in Mbits/s, and b and c are coefficients that depend on the 802.11 supplement. For 802.11a, $b = 161.5$ and $c = 156$. For 802.11b, $b = 754$ in the case of long preamble, or $b = 562$ in the case of short preamble, and $c = 112$.

But, when we share the medium with other users, the throughput drops significantly, as it was shown in Figure 4.10. We do not have a way to detect when other stations start using the radio channel. Therefore, to successfully be able to predict the resulting throughput, we implemented a throughput predictor emulator, which is a lookup table with throughput values for each rate setting, that we measured beforehand. The values are propagated as prediction values to the video codec, in the times when we know the radio channel is used by other stations. In such a way we can observe what the performance improvement will be in the ideal scenario of perfect throughput predictor.

As a practical implementation of a real throughput predictor, that takes into account the effects of medium sharing, we intend to investigate the following algorithm, as future work. Trying to transmit over the air is usually a bursty process (streaming, download), and the time between switching to active state and then back to inactive state is in the magnitude of seconds. Then, we can use the statistics for the observed throughput for each rate setting, during previous transmissions, as prediction what the real throughput will be during the next transmissions. As the medium utilization changes slowly, our prediction should work fine for the period that we are interested in (couple of tens of milliseconds).

4.3.5 Experiments

Our experimental setup is presented in Figure 4.11. The setup consists of two desktops (”golum” and ”quickbeam”) on an ethernet link, a 802.11a access point, and two laptops (”arwen” and ”eric”) – both running Linux. The laptops are equipped with 802.11a cards based on the Atheros AR5000 chipset, and the card driver uses the advanced hybrid rate control algorithm, as described in [Har05a]. The experiments were carried out by streaming a video file between ”arwen” and ”quickbeam” while ”arwen” was moving, following a predetermined track. The track consists of three parts - ”lead-in”, which is reaching from the room where the AP lies, to a specific start position in the hallway. Then follows moving with the laptop up and down the hallway for about 20s (”action”). Finally we go back (”lead-out”) again into the room. While the video streaming takes place, ”eric” downloads a file from ”gollum” (during the ”action” part) for about 10 seconds.

To evaluate the performance of the coupled rate-control systems, we have examined two cases. For each case, we have repeated the experiments several times. We managed to quite precisely time the experiments and to obtain a reasonable repeatability of the results.

1. **De-Coupled** : Decoupled rate control. The video codec has no indication of the actual throughput, and we set the target-rate for the VRCA to a fixed value, which is the *minimum* throughput that would be obtained in the case there is no medium sharing (about 4 Mbits/s - see Figure 4.10). In this way we cover both the cases when there is no cross-layer signaling (fixed rate) and when there is a throughput prediction feedback, but being incorrect due to the medium sharing.

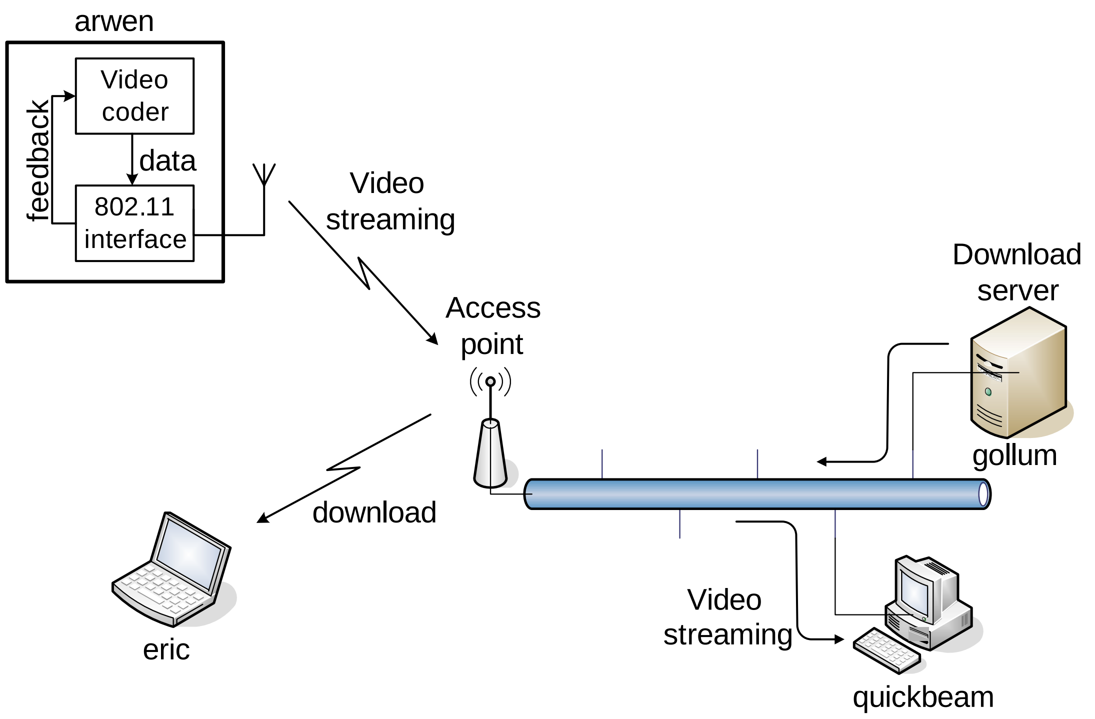


Figure 4.11: *Our experimental setup.*

2. **Coupled-MSP** : Coupled rate control, taking into account medium sharing. The rate control loops of the MAC-layer and the Video coder are coupled by letting the video coder obtain throughput prediction feedback information. In this case we are using the emulated throughput predictor that provides proper throughput estimation, when the medium is shared with another user.

In Figure 4.12 the quality (PSNR) is shown for the second ("action") part of the experiment[2]. In Figure 4.13 the received-signal strength is shown for the same time span, so the change in the link conditions due to the movement of "arwen" can be observed.

During the time when the download takes place, with the Coupled-MSP case we lose just a few frames, since the video encoder properly reduces the video-rate, following the throughput prediction feedback. With the De-Coupled experiment, the higher video-rate selected by the video encoder causes the wireless interface to choke during the time the download takes place, and the video freezes as a result of the multiple frame loss. This "freeze" can be localized by the regions with low PSNR. At about video frame 400, the video data manages to get through again, since the link conditions improve, as the latter can be observed on Figure 4.13 (the average SSIA increases).

[2]We have applied an averaging sliding window to the data for better visibility.

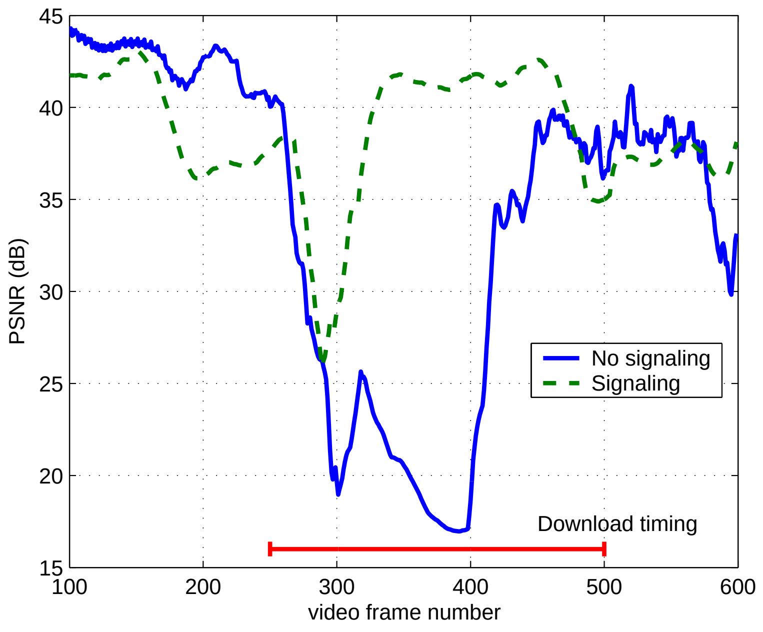


Figure 4.12: *PSNR of the received video: The solid line shows the results for a decoupled rate control using fixed rate (De-Coupled). The dashed line shows the quality during the experiment for the Coupled-MSP.*

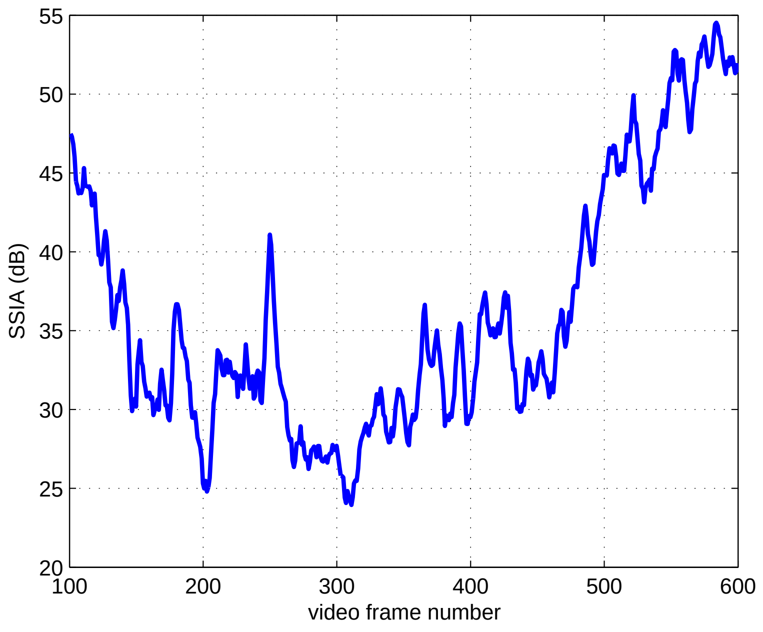


Figure 4.13: *Measured SSIA during experiments.*

The mean PSNR for the period of the whole De-Coupled experiment is 38.1dB, and for Coupled-MSP is 39.1dB. The mean PSNR for the period with download, for De-Coupled is 28.3dB, and for Coupled-MSP is 38.6dB. We see that a significant improvement (over 10dB) is achieved in the Coupled-MSP case, for the period the medium has been shared.

It is important to note that the results shown in Figure 4.12 are not the most optimistic in terms of improvement that we have obtained during our tests. This is because in many of the experiments with the De-Coupled case, the decoder on "quickbeam" would just stall because too many frames were lost, and consequently, we were not able to get the PSNR trace.

4.3.6 Conclusions and future work

In this section we have presented the results of a real-time streaming video experiment over a wireless 802.11a connection, in the presence of background traffic. We show that great improvements in the quality of the video can be achieved by cross-layer signaling, only in the case of correct estimation of the throughput decline due to the medium sharing. As future work, we plan to investigate practical implementation of a real throughput predictor, that takes into account the effects of medium sharing.

Chapter 5

Radio Abstraction Information Layer (RA*I*L)

The experiments reported in the previous chapter have proven that collaboration between streaming applications and the 802.11 MAC layer can provide significant quality improvements. Unfortunately, practice has shown that with QoS schemes/interfaces employing cross-layer interactions, a significant price is paid in terms of incompatibility of different proprietary inter-layer communication interfaces. Our goal is to define and build a unified interface (API) that can serve as intermediator between a wide variety of streaming applications and 802.11 equipment drivers. We call this interface RA*I*L - from Radio Abstraction Information Layer. Why it is **Radio** is obvious. The **Abstraction** part comes from the fact that the type of information we are communicating between layers is only of a kind that both layers understand well enough. **Information** suggests that the control communication is upwards only - i.e. informative for the application layer.

5.1 Design rationale

There are different control approaches in last-hop-wireless data networks to achieve QoS improvements when streaming applications are involved. We distinguish between three types of approaches based on the presence and the direction of control information flow between the application and the link network layers. The assumption we make here is that we have QoS-related control capabilities only at the application and the link layer. This is based on the following two considerations, which were already discussed earlier. The first is the well-known end-to-end QoS argument (see [Sal84]). The second is the argument that without link adaptation (which is a QoS-related control) there will be a number of times when no packets would go over the air at all. So, no matter how smart the network layers above are, this will have catastrophic consequences for the overall QoS of the system, especially for the cases with streaming applications involved.

We will make a short comparison of the three types of flow-control, in order to justify the structure and the capabilities of our RAIL API.

No cross-layer interactions - see Figure 5.1

This is the standard and the most widely used scheme. Every layer is "on its own". The application does not get any information from the Radio regarding the status of the link.

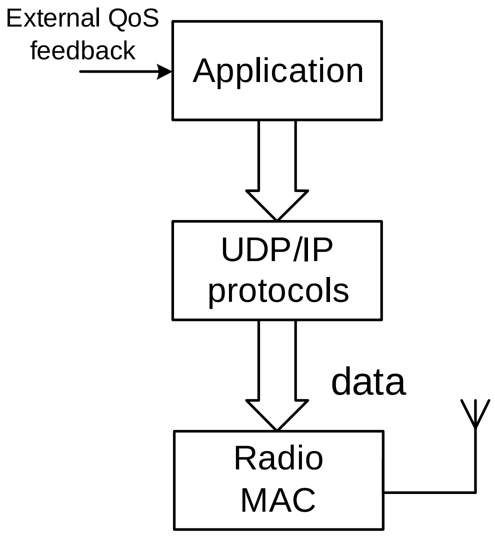


Figure 5.1: *No cross-layer interactions*

The application gets information about the data link quality though indirect methods, or by a feedback from the application at the other end. The major drawback of this approach is that often the feedback information comes too late at the application. Another problem is the additional control data overhead.

This scheme has the worst performance, because the quality of the feedback (if any) that an application gets is the worst, but the approach has the advantage of being simple.

Full control flow cross-layer interactions - see Figure 5.2

Here we have full interaction between layers. This means that on the one hand the application gets link quality information from the Radio, and on the other hand, the Radio gets "instructions" how to behave, or requests what link quality it should try to maintain. There are many variations

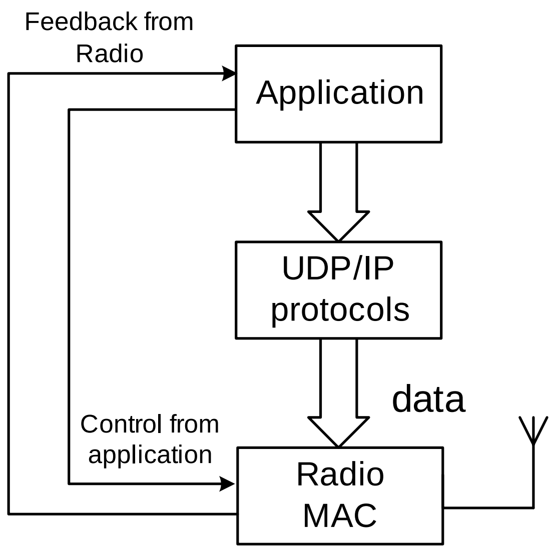


Figure 5.2: *Full control flow cross-layer interactions*

on the theme, including the most complicated scenario when actually series of QoS negotiations take place between layers before a "contract" is established (see [Dij00]).

This is the approach with potentially the best performance, but its major disadvantage is the big implementation and compatibility price to be paid. The latter is especially valid for the link layer where usually device drivers and firmware have to be specially designed or redesigned. Practice has shown that this will not happen, unless extremely necessary.

Cross-layer upflow-only interactions - see Figure 5.3

This is the case where the topmost layer only receives information from the bottom layer, but does not influence its decisions, neither tries to negotiate something. It is the method that we favor, and our API is designed with it in mind.

While this approach would give sub-optimal results compared with those of the previous scheme, it has the advantage of being "as simple as necessary, but not simpler". While almost completely avoiding the problem with compatibility and driver design efforts, our research shows that it achieves results very close to those achieved by the previous method. The reason for that is discussed in detail before. We also have determined that the information that matters most for the applications, and at the same time is universal between both layers, is the radio link throughput (or to be precise - goodput) and delay introduced by the radio layer. Also great improvements can be achieved if a prediction of the throughput, even if being

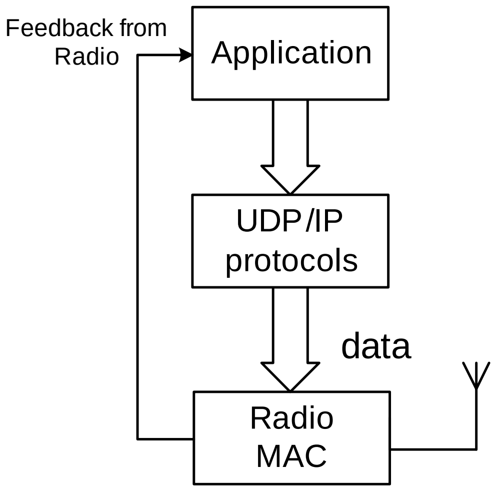


Figure 5.3: *Cross-layer interactions with control upflow only*

one for a very short time ahead, is provided. Therefore these are the main parameters that our API provides means to be communicated through.

5.2 RA*I*L API Reference

The API is designed as a supplement to the well-known Wireless Extensions for Linux, and is made as compatible as possible.

5.2.1 General usage information

There are two main types of information that an application can get from the Radio through RA*I*L. The first is statistics about throughput and packet delay, the other is throughput prediction. Before trying to obtain any information through RA*I*L, the application should inform itself about the statistics-gathering and prediction capabilities of the particular driver/firmware. This is done through SIOCGIWRAILSTATCAPA and SIOCGIWRAILPREDCAPA ioctls. Getting the statistics itself is done through SIOCGIWRAILSTAT ioctl. Depending on the capabilities of the radio firmware/driver (further referred to as Radio), average, minimum, or maximum values of throughput and/or delay statistics can be obtained. Also, if available, the statistics-gathering time window can be chosen by using SIOCSIWRAILSTATINTVL. Getting throughput predictions is done through SIOCGIWRAILPRED ioctl. There are two types of predicted values - standard, and extended. The extended throughput prediction is one that should be in general more accurate. It is up to the Radio developers to decide what type of prediction is supported, if any. Again, if available, the time ahead the prediction refers to can be chosen by using SIOCSIWRAILPREDINTVL.

5.2.2 IOCTLs

Statistics-related IOCTLs

SIOCGIWRAILSTATCAPA

Get the capabilities of the statistics-gathering module of the Radio.

Parameters

A pointer to a *iw_rail_capa* structure.

Remarks

This ioctl should be used along with SIOCGIWRAILPREDCAPA first - before any attempt to get statistics or prediction information.

SIOCGIWRAILSTAT

This ioctl is used by an application to retrieve the last collected statistics about the achieved throughput and packet delays.

Parameters

A pointer to a *iw_rail_stats* structure.

Remarks

Before calling the ioctl function, the *flags* member of *iw_rail_stats* should be set appropriately. See the description of the structure for more details. After the call, the application should check *status* member of *iw_rail_stats* to see if the statistics are valid and useable.

SIOCSIWRAILSTATINTVL

Set the time window, used by the Radio to calculate the throughput and delay statistics. This ioctl is subject of availability.

Parameters

A pointer to a __u32 variable containing the desired interval in milliseconds. The value should be in the interval $[min\_intvl\_ms, max\_intvl\_ms]$, as determined by calling SIOCGIWRAILSTATCAPA.

Remarks

The information from a previous SIOCGIWRAILSTATCAPA ioctl call should be used to determine if this ioctl is permitted.

SIOCSIWRAILSTATRESET

Reset the state of the statistics module in the Radio, and clear all the buffers/values related to statistics calculation.

Parameters

A pointer to a __u32 variable indicating what to reset. It contains a combination of the following flags:

Flag	Description
IW_RAIL_STATSR_THR	Reset throughput statistics
IW_RAIL_STATSR_DLY	Reset delay statistics
IW_RAIL_STATSR_INTVL	Reset interval to default

Remarks

Only the flags that correspond to features that the Radio is capable of (obtainable through SIOCGIWRAILSTATCAPA ioctl) should be used. Also see the description of *iw_rail_capa* structure for more information about default intervals.

Prediction-related IOCTLs

SIOCGIWRAILPREDCAPA

Get the capabilities of the throughput prediction module of the Radio.

Parameters

A pointer to a *iw_rail_capa* structure.

Remarks

This ioctl should be used along with SIOCGIWRAILSTATCAPA first - before any attempt to get statistics or prediction information.

SIOCGIWRAILPRED

This ioctl is used by an application to get a throughput prediction.

Parameters

A pointer to a *iw_rail_preds* structure.

Remarks

Before calling the ioctl function, the *pkt_size* member of *iw_rail_preds* should be set to the desired packet size that the prediction should be made for. See the description of the structure for more details. After the call, the application should check *status* member of *iw_rail_preds* to see if the predicted value(s) are valid and useable.

SIOCSIWRAILPREDINTVL

Set the time ahead, that the prediction information provided by the Radio refers to. This ioctl is subject of availability.

Parameters

A pointer to a __u32 variable containing the desired interval in milliseconds. The value should be in the interval [*min_intvl_ms*, *max_intvl_ms*], as determined by calling SIOCGIWRAILPREDCAPA.

Remarks

The information from a previous SIOCGIWRAILPREDCAPA ioctl call should be used to determine if this ioctl is permitted.

SIOCSIWRAILPREDRESET

Reset the state of the prediction module in the Radio.

Parameters

A pointer to a __u32 variable indicating what to reset. It contains a combination of the following flags:

Flag	Description
IW_RAIL_PREDR_PRE	Reset throughput predictor
IW_RAIL_PREDR_INTVL	Reset prediction interval to default

Remarks

Using IW_RAIL_PREDR_INTVL makes sense only if the Radio allows for setting the prediction interval. This information is obtainable through SIOCGIWRAILPREDCAPA ioctl. Also see the description of *iw_rail_capa* structure for more information about default intervals.

5.2.3 Data structures

iw_rail_stats

This structure is used with the SIOCGIWRAILSTAT ioctl for getting the statistics information.

Syntax

```
struct iw_rail_stats {
    /* input params */
    __u16       flags;      /* type of throughput and
                               delay to get */
    /* output params */
    __u32       thr;        /* Throughput */
    __u32       delay;      /* delay */
    __u16       status;     /* validity of the statistics */
    __u32       time_ms;    /* timestamp in ms */
    __u16       dimensions; /* delay and throughput
                               dimensions */
};
```

Members

flags [in]

Type of throughput and delay statistics to get. It is a logical combination of a throughput and a delay capability flag:

Type	Flag	Description
Throughput flags	IW_RAIL_STATSG_THR_AV	Get average throughput
	IW_RAIL_STATSG_THR_MN	Get minimum throughput
	IW_RAIL_STATSG_THR_MX	Get maximum throughput
Delay flags	IW_RAIL_STATSG_DLY_AV	Get average delay
	IW_RAIL_STATSG_DLY_MN	Get minimum delay
	IW_RAIL_STATSG_DLY_MX	Get maximum delay

thr [out]

Throughput statistics in dimension specified by the *dimensions* member.

delay [out]

Delay statistics in dimension specified by the *dimensions* member. Delay is specified as the time between the moment a packet is delivered by an upper layer to be transmitted by the Radio, to the moment it gets received by the other station (which is acknowledged by an ACK message).

status [out]

Validity of the throughput and the delay statistics. It is possible that sometimes, for example because of lack of enough data transmitted, the statistics will not to be able to be calculated properly. An application can check this member to determine what part of the statistics is valid and what not. The value is a combination of a delay and a throughput validity flag:

Type	Flag	Description
Throughput flags	IW_RAIL_STATSS_THR_VALID	The throughput value is valid
	IW_RAIL_STATSS_THR_INVALID	The throughput value is invalid/unreliable
Delay flags	IW_RAIL_STATSS_DLY_VALID	The delay value is valid
	IW_RAIL_STATSS_DLY_INVALID	The delay value is invalid/unreliable

time_ms [out]

Timestamp when the statistics were last updated, in milliseconds.

dimensions [out]

Dimensions of the delay and throughput values. It is a combination of a delay and a throughput dimension flag:

Type	Flag	Description
Throughput flags	IW_RAIL_THR_MBIT	The throughput value is in Mbits/s
	IW_RAIL_THR_KBIT	The throughput value is in Kbits/s
Delay flags	IW_RAIL_DLY_MS	The delay value is in milliseconds
	IW_RAIL_DLY_US	The delay value is in microseconds

iw_rail_preds

This structure is used with the SIOCGIWRAILPRED ioctl for getting the throughput prediction information.

Syntax

```
struct iw_rail_preds {
    /* input params */
    __u16      pkt_size;   /* packet size to use for
                              prediction calculation */
    /* output params */
    __u32      thr;        /* predicted throughput */
    __u32      expert_thr; /* "Expert" predicted throughput */
    __u16      status;     /* validity of the predictions */
    __u32      time_ms;    /* (future) prediction timestamp
                              in ms */
    __u16      dimension;  /* throughput dimension */
};
```

Members

pkt_size [in]

The packet size to use when calculating the throughput prediction. This should include the UDP/IP header overhead as well, but is not critical, since this overhead is usually much lower than the useable packet sizes.

thr [out]

Throughput prediction value in dimension specified by the *dimension* member. This is raw throughput prediction - it can be pretty inaccurate.

expert_thr [out]

"Expert" throughput prediction value in dimension specified by the *dimension* member. This is a more sophisticated throughput prediction that takes into account different additional factors like medium sharing, etc. Thus, it is expected to be more accurate than *thr* as well. Providing this value is optional - an application should check the driver prediction capabilities by using the SIOCGIWRAILPREDCAPA ioctl before using it.

status [out]

Validity of the throughput and the delay predictions. It is possible that sometimes, for example because of lack of enough statistics, the predictions not to be able to be calculated properly. An application can check this member to determine what part of the predictions is valid and what not. The value is a combination of a throughput and an extended throughput validity flag:

Type	Flag	Description
Throughput flags	IW_RAIL_PREDS_THR_VALID	The throughput value is valid
	IW_RAIL_PREDS_THR_INVALID	The throughput value is invalid/unreliable
Delay flags	IW_RAIL_PREDS_ETHR_VALID	The extended throughput value is valid
	IW_RAIL_PREDS_ETHR_INVALID	The extended throughput value is invalid/unreliable

time_ms [out]

Timestamp referring to the future moment the prediction applies to, in milliseconds.

dimension [out]

Dimension of the throughput prediction values. It uses the same throughput flags as in *dimensions* member of *iw_rail_stats*.

iw_rail_capa

This is the structure used with the *CAPA ioctls for getting the capabilities.

Syntax

```
 struct iw_rail_capa {
    __u16       flags;          /* capabilities flags */
    __u32       max_intvl_ms;   /* maximum interval in ms */
    __u32       min_intvl_ms;   /* minimum interval in ms */
    __u32       deflt_intvl_ms; /* default interval in ms */
};
```

Members

flags [out]

Capabilities regarding statistics collection and prediction. For the statistics collection capabilities, the following flags are used:

Type	Flag	Description
Throughput flags	IW_RAIL_STATSC_THR_AV	Can get average throughput
	IW_RAIL_STATSC_THR_MN	Can get minimum throughput
	IW_RAIL_STATSC_THR_MX	Can get maximum throughput
Delay flags	IW_RAIL_STATSC_DLY_AV	Can get average delay
	IW_RAIL_STATSC_DLY_MN	Can get minimum delay
	IW_RAIL_STATSC_DLY_MX	Can get maximum delay
Interval flags	IW_RAIL_STATSC_INTVL	Can set interval

For the prediction capabilities, the following flags are used:

Flag	Description
IW_RAIL_PREDC_THR	Can get throughput
IW_RAIL_PREDC_ETHR	Can get extended throughput
IW_RAIL_PREDC_INTVL	Can set prediction interval

max_intvl_ms [out]

The maximum time interval in milliseconds that can be set for the operation in question (statistics or prediction).

min_intvl_ms [out]

The minimum time interval in milliseconds that can be set for the operation in question (statistics or prediction).

deflt_intvl_ms [out]

The default time interval in milliseconds that is implicitly set for the operation in question (statistics or prediction). For the statistics this interval is the standard interval the driver is designed to calculate/keep statistics for. It often is associated with the Link Adaptation part of the work of the driver and assures the best results in terms of resource usage and statistics accuracy. For the prediction this member gives the optimal time interval that the driver makes the throughput prediction ahead. This optimal interval is the best tradeoff between accuracy of the prediction and maximum look-ahead.

5.3 Full list of the RA*I*L code

```
/* RAIL interface ioctls */
/* statistics-related ioctls */
#define SIOCGIWRAILSTAT         0x8B80  /* get applications
                                           specific stats */
#define SIOCGIWRAILSTATCAPA     0x8B81  /* get statistics
                                           capabilities */
#define SIOCSIWRAILSTATINTVL    0x8B82  /* set stats averaging
                                           interval */
#define SIOCSIWRAILSTATRESET    0x8B83  /* reset statistics */

/* prediction-related ioctls */
#define SIOCGIWRAILPRED         0x8B88  /* get apps specific
                                           predictions */
#define SIOCGIWRAILPREDCAPA     0x8B89  /* get predictions
                                           capabilities */
#define SIOCSIWRAILPREDINTVL    0x8B8A  /* set prediction interval */
#define SIOCSIWRAILPREDRESET    0x8B8B  /* reset predictors */

/* RAIL constants */
/* Statistics capabilities - in flags */
#define IW_RAIL_STATSC_THR_AV   0x0001  /* Can get average thr-put */
#define IW_RAIL_STATSC_THR_MN   0x0002  /* Can get minimum thr-put */
#define IW_RAIL_STATSC_THR_MX   0x0004  /* Can get maximum thr-put */
#define IW_RAIL_STATSC_DLY_AV   0x0010  /* Can get average delay */
#define IW_RAIL_STATSC_DLY_MN   0x0020  /* Can get minimum delay */
#define IW_RAIL_STATSC_DLY_MX   0x0040  /* Can get maximum delay */
#define IW_RAIL_STATSC_INTVL    0x0100  /* Can set interval */

/* status of the statistics */
#define IW_RAIL_STATSS_THR_VALID   0x0001 /* The throughput
                                             value is valid */
#define IW_RAIL_STATSS_THR_INVALID 0x0002 /* The throughput value
                                             is invalid/unreliable */
#define IW_RAIL_STATSS_DLY_VALID   0x0010 /* The delay value
                                             is valid */
#define IW_RAIL_STATSS_DLY_INVALID 0x0020 /* The delay value is
                                             invalid/unreliable */

/* status of the predictions */
#define IW_RAIL_PREDS_THR_VALID    0x0001 /* The throughput value
                                             is valid */
#define IW_RAIL_PREDS_THR_INVALID  0x0002 /* The throughput value
                                             is invalid/unreliable */
```

```
#define IW_RAIL_PREDS_ETHR_VALID   0x0010 /* The extended throughput
                                              value is valid */
#define IW_RAIL_PREDS_ETHR_INVALID 0x0020 /* The extended throughput
                                        value is invalid/unreliable */

/* Statistics reset */
#define IW_RAIL_STATSR_THR      0x0001  /* Reset throughput */
#define IW_RAIL_STATSR_DLY      0x0002  /* Reset delay */
#define IW_RAIL_STATSR_INTVL    0x0004  /* Reset interval to dflt */

/* Type of throughput statistics to get */
#define IW_RAIL_STATSG_THR_AV   0x0001  /* Get average throughput */
#define IW_RAIL_STATSG_THR_MN   0x0002  /* Get minimum throughput */
#define IW_RAIL_STATSG_THR_MX   0x0003  /* Get maximum throughput */
/* Type of delay statistics to get */
#define IW_RAIL_STATSG_DLY_AV   0x0010  /* Get average delay */
#define IW_RAIL_STATSG_DLY_MN   0x0020  /* Get minimum delay */
#define IW_RAIL_STATSG_DLY_MX   0x0030  /* Get maximum delay */

/* Predictors capabilities - in flags */
#define IW_RAIL_PREDC_THR       0x0001  /* Can get throughput */
#define IW_RAIL_PREDC_ETHR      0x0002  /* Can get extended thr-put */
#define IW_RAIL_PREDC_INTVL     0x0004  /* Can set prediction intvl */

/* Predictors reset */
#define IW_RAIL_PREDR_PRE       0x0001  /* Reset predictors */
#define IW_RAIL_PREDR_INTVL     0x0002  /* Reset intvl to default */

/* Throughput dimension */
#define IW_RAIL_THR_MBIT        0x0001  /* Mbits/s */
#define IW_RAIL_THR_KBIT        0x0002  /* Kbits/s */
/* Delay dimension */
#define IW_RAIL_DLY_MS          0x0010  /* milliseconds */
#define IW_RAIL_DLY_US          0x0020  /* microseconds */

/* RAIL structures */

/* for statistics */
struct iw_rail_stats {
    /* input params */
    __u16       flags;      /* type of throughput and delay to get */
    /* output params */
    __u32       thr;        /* Throughput */
    __u32       delay;      /* delay */
    __u16       status;     /* validity of the statistics */
    __u32       time_ms;    /* timestamp in ms */
    __u16       dimensions; /* delay and throughput dimensions */
};

/* for predictions */
struct iw_rail_preds
{
    /* input params */
    __u16       pkt_size;   /* packet size to use for prediction
                               calculation */
    /* output params */
    __u32       thr;        /* predicted throughput */
    __u32       expert_thr; /* "Expert" predicted throughput */
    __u16       status;     /* validity of the predictions */
    __u32       time_ms;    /* prediction timestamp in ms */
    __u16       dimension;  /* throughput dimension */
};

/* for capabilities */
struct iw_rail_capa
{
    __u16       flags;          /* capabilities flags */
    __u32       max_intvl_ms;   /* maximum interval in ms */
    __u32       min_intvl_ms;   /* minimum interval in ms */
    __u32       deflt_intvl_ms; /* default interval in ms */
};
```

5.4 Conclusions

In this chapter, we have presented RA*I*L, an API that assists the Application-layer in adapting to network performance changes due to variations in the quality of the radio link. The application is informed about the current achieved throughput by the wireless network interface. It can also obtain a short-time prediction of the throughput (if supported by the link layer), which is a great help in the case of a streaming application for example.

Using an unified, standardized API, brings compatibility between software components and greatly improves the efficiency and the speed of the design process. In our case, RA*I*L brings control communication compatibility between components lying in different network layers, such as drivers (that are on the link/transport layer) and applications. The result would be better spreading of cross-layer signaling schemes, that usually see no large deployment due to compatibility-related problems.

Chapter 6

Conclusions

THIS thesis addressed the ever-persistent problem of quality guarantees in wireless data networks. The quality guarantees are especially important for real-time streaming applications. While in fixed links most of the quality issues could be, and are still resolved by over-provisioning, in the world of wireless communications things get complicated. The main reason for wireless links being a bottleneck in terms of both achievable throughput and packet loss and delay guarantees, is the resource limitation that will always be there. This limitation has two aspects. The first aspect is the constrained amount of power that can be emitted in the air, both because mobile devices do not have powerful energy sources, and because more radio power results in more interference to the other users. This power constraint consequently limits range and leads to a significant number of situations of unsuccessful transmissions due to a weak signal. The second aspect is that the radio spectrum is a shared resource, meaning that there are unsuccessful transmissions due to interference. Unsuccessful transmissions result in packet losses and increased packet delays because of the need of retransmissions.

6.1 Approach

Our work showed that while there is no magic solution that can resolve all link quality-related problems in wireless networks, a lot can be done to mitigate them.

The first, and the most important, is to make the radio as adaptive to the changes of the link conditions as possible. This can be done through our advanced hybrid rate-control algorithm. The algorithm combines a stable throughput-based solution with a rapid link-quality feedback supplement into a novel controller that has both swift response and stable performance. An important feature of this solution is that it does not require changes of the 802.11 standard, as much other similar work does. Therefore the cost of implementation, which is changes in drivers or in chipsets (that are getting updated every day anyway), is low, and makes the benefit-to-price ratio high.

The second solution, which is dependant on the first one, is the propagation of link status-related information from the link layer to the application. The latter can use this feedback to change its data stream properties to match those of the underlying wireless link. This sort of behaviour proves most useful in cases when the radio cannot compensate for a deterioration of the radio connection such as a significant drop of the signal strength due to obstruction, or too many users competing to use the same medium. In those situations the standard outcome is bad behaviour of streaming applications, for example stalling video or video with artifacts. Both are very annoying to humans. Our cross-layer communication scheme helps avoiding drop-outs by trading them for slightly degraded picture quality (more compression), which is much better appreciated.

A good representation of our final completed system is shown on Figure 6.1.

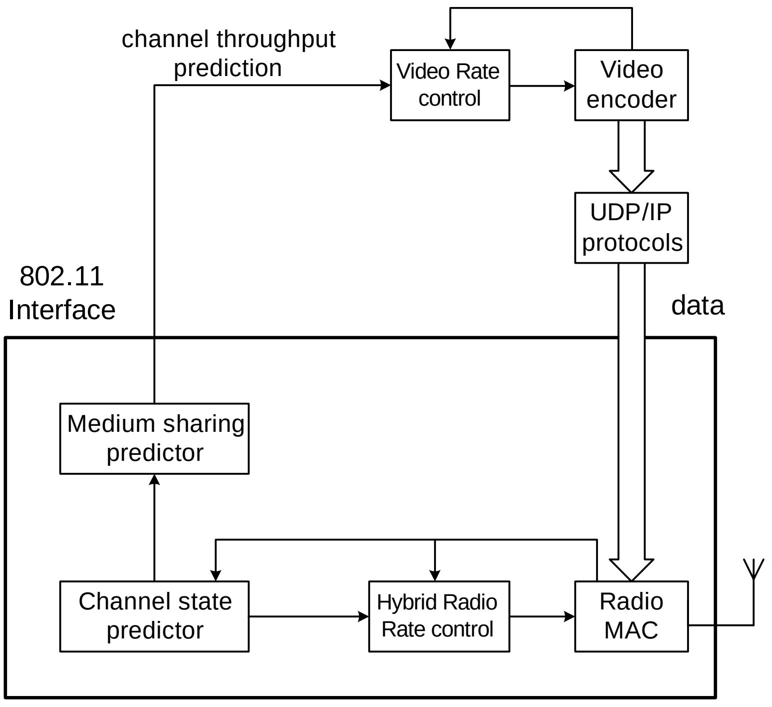


Figure 6.1: *The final structure of our system*

6.2 Results

The experiments presented in Chapter 3 show that our LA solution has a swift response to rapid changes in link quality. This translates in very low packet latency and packet loss, especially when compared with traditional LA approaches under similar conditions. The result is significant improvement in quality delivered by streaming applications.

We have shown in Chapter 4 that even more quality increase is indeed gained by giving information to the application about the current and the future status of the wireless link.

We discovered that the most beneficial approach employing cross-layer control information exchange is a bottom-up only informational policy. In other words, the improvement that any more complicated scheme, like one that also includes some degree of control from the application over the radio (top down supervision), is limited, while the price paid in terms of design efforts and compatibility is high.

An important finding of our work is also the great significance of the accuracy of the feedback that goes from the radio to the application. More precisely, if that feedback does not accurately enough present the current (and future) characteristics of the radio link, the final result is usually worse than the case without any cross-layer feedback at all.

6.3 Suggestions for future work

There are several things that can serve as enhancement to our work.

A more theoretical study of the control model of the radio interface would be a valuable extension to the work presented in this thesis. A valid question arising from Chapter 3, is how to determine the parameters of the rate-control algorithm. This was initially done by applying some control theory knowledge. However engineers building wireless communication systems will most likely do not have such, or even similar, background. Therefore, clear rules or an algorithm for tuning these parameters, would be very beneficial. A mathematical model that would help build such an algorithm would then be a good follow-up of our work.

Another possible future direction could be a more detailed study of realistic cases when there are multiple users in a Basic Service Set (BSS). A good continuation of this research would be a practical implementation and validation of the medium sharing predictor described in Chapter 4.

Bibliography

[Agi01] Agilent Technologies. IEEE 802.11 Wireless LAN PHY Layer (RF) Operation and Measurement. Application note 1380-2, Apr 2001.

[Aya95] E. Ayanoglu, S. Paul, T. LaPorta, K. Sabnani, and R. Gitlin. AIRMAIL: A Link-Layer Protocol for Wireless Networks. *ACM Wireless Networks*, vol. 1(1):pp. 47–60, 1995.

[Bak95] A. Bakre and B. Badrinath. I-TCP: Indirect TCP for mobile hosts. *15th International Conference on Distributed Computing Systems*, 1995.

[Bal95] H. Balakrishnan, S. Seshan, and R. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, vol. 1(4), 1995.

[Bal99] K. Balachandran, S. Kadaba, and S. Nanda. Channel quality estimation and rate adaptation for cellular mobile radio. *IEEE Journal on Selected Areas in Communications*, vol. 17(7):pp. 1244–1256, Jul 1999.

[Bal02] R. Balan, B. Lee, K. Kumar, L. Jacob, W. Seah, and A. Ananda. TCP HACK: TCP Header Checksum Option to Improve Performance Over Lossy Links. *Computer Networks*, vol. 39(4):pp. 347–361, 2002.

[Bia00] G. Bianchi. Performance analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, vol. 18(3):pp. 535–547, Mar 2000.

[Bol98] J.-C. Bolot and T. Turletti. Experience with control mechanisms for packet video in the internet. *Computer Communication Review*, 1998.

[Bra01] B. Braswell and J. McEachen. Modeling Data Rate Agility in the IEEE 802.11a WLAN Protocol. In *OPNETWORK 2001*. Mar 2001.

[Cha96] H. Chaskar, T. Lakshman, and U. Madhow. On the design of interfaces for tcp/ip over wireless. In *Proceedings of MILCOM '96 Conference*. Oct 1996.

[Che03] P. Chevillat, J. Jelitto, A. Barreto, and H. Truong. A dynamic link adaptation algorithm for IEEE 802.11a wireless LANs. In *Proceedings of IEEE International Conference on Communications (ICC'03)*. May 2003.

[Che04a] H. Chen and Y. Li. Performance model of IEEE 802.11 DCF with variable packet length. *IEEE Communications Letters*, vol. 8(3):pp. 186–188, Mar 2004.

[Che04b] K. Chen, Y. Xue, S. Shah, and K. Nahrstedt. Understanding bandwidth-delay product in mobile ad hoc networks. *Elsevier Computer Communications*, vol. 27(10):pp. 923–934, 2004.

[Cla94] A. Claessen, L. Monteban, and H. Moelard. The AT&T GIS WaveLAN air interface and protocol stack. In *Proceedings of 5th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. Sep 1994.

[Cro97] B. Crow, I. Widjaja, J. Kim, and P. Sakai. Investigation of the IEEE 802.11 medium access control (MAC) sublayer functions. In *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '97)*. Apr 1997.

[DeS93] A. DeSimone, M. Chuah, and O. Yue. Throughput performance of transport-layer protocols over wireless lans. In *Proceedings of IEEE Globecom '93*. Nov 1993.

[DH00] A. Duel-Hallen, S. Hu, and H. Hallen. Long-range prediction of fading signals. *IEEE Signal Processing Magazine*, vol. 17(3):pp. 62–75, May 2000.

[Dij00] H. van Dijk, K. Langendoen, and H. Sips. ARC: a bottom-up approach to negotiated QoS. In *3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000)*, pp. 128–137. Monterey, CA, Dec 2000.

[Fit98] F. Fitzek, B. Rathke, M. Schläger, and A. Wolisz. Quality of Service Support for Real-Time Multimedia Apllications over Wireless Links using the Simultaneous MAC-Packet Transmission (SMTP) in a CDMA Environment. In *Proceedings of MoMuc '98*. Oct 1998.

[Gri03] A. Grilo and M. Nunes. Link-adaptation and transmit power control for unicast and multicast in IEEE 802.11 a/h/e WLANs. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN '03)*. Oct 2003.

[Gun02] M. Güneş and D. Vlahovic. The Performance of the TCP/RCWE Enhancement for Ad-Hoc Networks. In *Proceedings of 7th IEEE Symposium on Computers and Communications (ISCC)*. Jul 2002.

[Gup00] P. Gupta and P. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, vol. 46(2):pp. 388–404, Mar 2000.

[Han97] L. Hanzo. The pan-european cellular system. In J. Gibson, editor, *The Communications Handbook*, pp. 1239 – 1241. CRC press, 1997.

[Har03] I. Haratcherev, K. Langendoen, I. Lagendijk, and H. Sips. Application-Directed Automatic 802.11 Rate Control: Design Rationale and Preliminary Results. In *ASCI 2003*, pp. 56–60. Heijen, The Netherlands, Jun 2003.

[Har04] I. Haratcherev, K. Langendoen, I. Lagendijk, and H. Sips. Hybrid Rate Control for IEEE 802.11. In *ACM International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, pp. 10–18. Philadelphia, PA, USA, Oct 2004.

[Har05a] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. Automatic IEEE 802.11 rate control for streaming applications. *Wireless Communications and Mobile Computing, special issue on Radio Link and Transport Protocol Engineering for Future-Generation Wireless Mobile Data Networks*, vol. 5(4):pp. 421–437, Jun 2005.

[Har05b] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, and H. Sips. Fast 802.11 link adaptation for real-time video streaming by cross-layer signaling. In *Proc. International Symposium on Circuits and Systems*. Kobe, Japan, May 2005.

[Heu03] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, pp. 836–843. San Francisco, California, USA, Apr 2003.

[Hol01] G. Holland, N. H. Vaidya, and P. Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *ACM MOBICOM'01*. Rome, Italy, Jul 2001.

[Hu02] Y.-C. Hu and D. Johnson. Design and demonstration of live audio and video over multihop wireless ad hoc networks. In *Proceedings of MILCOM 2002*. Oct 2002.

[IEEwg] IEEE. Quick Guide To IEEE 802.11 Work Groups and Activities. `http://grouper.ieee.org/groups/802/11/QuickGuide_IEEE_802_WG_and_Activities.htm`.

[IEE11] IEEE standard 802.11. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[IEE11a] IEEE standard 802.11a supplement. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. High-speed Physical Layer in the 5 GHz Band, 1999.

[IEE11b] IEEE standard 802.11b supplement. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Higher-speed Physical Layer in the 2.4 GHz Band, 1999.

[IEE11h] IEEE standard 802.11h supplement. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Spectrum and Transmit Power Management Extensions in the 5GHz band in Europe, 2003.

[IEE11e] IEEE standard 802.11e supplement. MAC Enhancements for Quality of Service. `http://grouper.ieee.org/groups/802/11/Reports/tge_update.htm`, 2005.

[Int97] Intersil Corporation. Tutorial on Basic Link Budget Analysis. Application note AN9804.1, Jun 1997.

[Int99] Intersil Corporation. Brief Tutorial on IEEE 802.11 Wireless LANs. Application note AN9829, Feb 1999.

[Int00a] Intersil Corporation. A Condensed Review of Spread Spectrum Techniques for ISM Band Systems. Application note AN9820.1, May 2000.

[Int00b] Intersil Corporation. Complementary Code Keying Made Simple. Application note AN9850.1, May 2000.

[Jea96] Jean Tourrilhes. Linux Wireless Extensions. `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`, 1996.

[Jia01] H. Jiang, S. Cheng, and X. Chen. TCP Reno and Vegas Performance in Wireless Ad Hoc Networks. In *Proceedings of IEEE International Conference on Communications*. Jun 2001.

[Jun03] J. Jun, P. Peddabachagari, and M. L. Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. In *Second IEEE International Symposium on Network Computing and Applications*, pp. 249–256. Cambridge, Massachusetts, USA, Apr 2003.

[Kam97] A. Kamerman and L. Monteban. WaveLAN-II: A High-Performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, pp. 118–133, Summer 1997.

[Kle05] F. Klemm, Z. Ye, S. Krishnamurthy, and S. Tripathi. Improving TCP performance in ad hoc networks using signal strength based link management. *Elsevier Ad Hoc Networks*, vol. 3(2):pp. 175–191, Aug 2005.

[Lac04] M. Lacage, M. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proceedings of ACM International Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)*. Oct 2004.

[Lam02] M. Lampe, H. Rohling, and W. Zirwas. Misunderstandings about link adaptation for frequency selective fading channels. In *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*. Sep 2002.

[Leu01] K. Leung, P. Driessen, K. Chawla, and X. Qiu. Link Adaptation and Power Control for Streaming Services in EGPRS Wireless Networks. *IEEE Journal on Selected Areas in Communications*, vol. 19(10):pp. 2029–2039, Oct 2001.

[Lev96] W. Levine, editor. *The Control Handbook*. CRC press, 1996.

[Lin00] Z. Lin, G. Malmgren, and J. Torsner. System performance analysis of link adaptation in HiperLAN type 2. In *Proceedings of IEEE Vehicular Technology Conference (VTC 2000)*. 2000.

[Mil97] L. Milstein and M. Simon. Spread spectrum communications. In J. Gibson, editor, *The Communications Handbook*, pp. 199 – 212. CRC press, 1997.

[Nun04] T. Nunome and S. Tasaka. An application-level QoS comparison of single-stream and multi-stream approaches in a wireless ad hoc network. In *Proceedings of 13th International Conference on Computer Communications and Networks*. 2004.

[Oba04] M. Obaidat and D. Green. An adaptive protocol model for IEEE 802.11 wireless LANs. *Elsevier Computer Communications*, vol. 27(12):pp. 1131–1136, 2004.

[Ort95] A. Ortega and M. Khansari. Rate control for video coding over variable bit rate channels with applications to wireless transmission. In *Proceedings of the 2nd IEEE International Conference on Image Processing (ICIP)*. Oct 1995.

[Pah95] K. Pahlavan, T. Probert, and M. Chase. Trends in Local Wireless Networks. *IEEE Communications Magazine*, vol. 33(3):pp. 88–95, Mar 1995.

[Pil03] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha. Understanding TCP fairness over Wireless LAN. In *Proceedings of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM).* Mar 2003.

[PP03] J. del Prado Pavon and S. Choi. Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement. In *IEEE International Conference on Communications, 2003 (ICC '03)*, vol. 2, pp. 1108–1113. Anchorage, Alaska, USA, May 2003.

[Pra01] A. Prasad. Performance Evaluation, System Design and Network Deployment of IEEE 802.11. *ACM Wireless Personal Communications*, vol. 19:pp. 57–79, 2001.

[Qia02] D. Qiao, S. Choi, and K. G. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *IEEE Transactions on Mobile Computing*, vol. 1(4):pp. 278–292, 2002.

[Que99] O. Queseth, F. Gessler, and M. Frodigh. Algorithms for Link Adaptation in GPRS. In *Proceedings of IEEE Vehicular Technology Conference (VTC '99).* May 1999.

[Sal84] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, vol. 2(4):pp. 277–288, Nov 1984.

[Sch01] H. Schulzrinne. 20 Years Old and Ready to Get a Job? `http://www.cs.columbia.edu/~hgs/papers/2001/iwqos.pdf`, Jun 2001. invited talk, IWQoS.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, vol. 27:pp. 379–423, 623–656, 1948.

[Sin02] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. WTCP: A reliable transport protocol for wireless wide-area networks. *Wireless Networks*, vol. 8(2-3):pp. 301–316, 2002.

[Spa00] Z. Spasojevic and J. Burns. Performance Comparison of Frequency Hopping and Direct Sequence Spread Spectrum Systems in the 2.4 GHz Range. In *Proceedings of the 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2000).* Sep 2000.

[Sun04] K. Sundaresan, H.-Y. Hsieh, and R. Sivakumar. IEEE 802.11 over multi-hop wireless networks: problems and new perspectives. *Elsevier Ad Hoc Networks*, vol. 2(2):pp. 109–132, Apr 2004.

[Taa02] J. Taal, K. Langendoen, A. van der Schaaf, H. van Dijk, and R. Lagendijk. Adaptive end-to-end optimization of mobile video streaming using QoS negotiation. In *ISCAS, special session on Multimedia over Wireless Networks*, vol. I, pp. 53–56. Scottsdale, AZ, May 2002.

[Taa03] J. Taal, I. Haratcherev, K. Langendoen, and R. Lagendijk. Quality of service controlled adaptive video coding over IEEE 802.11 wireless links. In *ICME, special session on Networked Video*. Baltimore, MD, Jun 2003.

[Tay01] Y. Tay and K. Chua. A capacity analysis for the IEEE 802.11 MAC protocol. *ACM Wireless Networks*, vol. 7(2):pp. 159–171, 2001.

[Ue98] T. Ue, S. Sampei, N. Morinaga, and K. Hamaguchi. Symbol rate and modulation level-controlled adaptive modulation/TDMA/TDD system for high-bit-rate wireless data transmission. *IEEE Transactions on Vehicular Technology*, vol. 47(4):pp. 1134–1147, Nov 1998.

[Veg02] A. van der Vegt. Auto Rate Fallback Algorithm for the IEEE 802.11a Standard. Tech. rep., Utrecht University, 2002.

[Wik06] Wikipedia, the free encyclopedia. Link Adaptation. `http://en.wikipedia.org/wiki/Link_adaptation`.

[Wil02] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz. Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics*, vol. 49(6):pp. 1265–1282, Dec 2002.

[Xia99] X. Xiao and L. M. Ni. Internet QoS: A Big Picture. *IEEE Network*, vol. 13(2):pp. 8–18, Mar 1999.

[Zha00] W. Zhao, D. Olshefski, and H. Schulzrinne. Internet Quality of Service: an Overview, 2000. http://www.cs.columbia.edu/ hgs/netbib/.

Index

Samenvatting

Het onderwerp dat in dit proefschrift aan de orde komt is de onmogelijkheid om kwaliteitsgaranties te bieden in draadloze netwerk verbindingen. In tegenstelling tot communicatie over vaste (bedraadde) netwerken waar overcapaciteit voldoende soelaas biedt om fluctuaties in de hoeveelheid dataverkeer op te vangen, heeft communicatie in draadloze netwerken te maken met fundamentele resource beperkingen die leiden tot lage transport snelheden, data verlies en (onbegrensde) vertragingen. We onderscheiden twee soorten resource beperkingen. Ten eerste is het uitgezonden zendvermogen gelimiteerd omdat draagbare apparatuur nu eenmaal gevoed wordt door een accu (batterij) met beperkte capaciteit. Het gevolg is dat de uitgezonden signalen relatief zwak zijn hetgeen de maximale afstand tussen zender en ontvanger beperkt en tevens binnen dit bereik een groot aantal posities oplevert waar correcte ontvangst onmogelijk is. Ten tweede wordt het radio spectrum gedeeld door meerdere gebruikers die elkaars signalen negatief beïnvloeden (interferentie) als ze vlak bij elkaar gepositioneerd zijn. Om deze interferentie te minimaliseren is het noodzakelijk het zendvermogen te beperken hetgeen de ontvangst-kwaliteit weer vermindert. De resulterende fouten in het dataverkeer kunnen gecompenseerd worden mbv. herhaald zenden van boodschappen (retransmissions) hetgeen vertragingen oplevert en de effectieve snelheid over de draadloze verbinding verlaagt.

De problemen met draadloze communicatie hebben vooral een nadelig effect op streaming applicaties die strikte real-time eisen hebben om goed te functioneren; er is niets vervelender dan een haperende video conferentie waar beelden voortdurend even 'bevriezen' ten gevolge van één of meerdere retransmissions. Ondanks de fundamentele aard van de problemen laat de in dit proefschrift gepresenteerde aanpak zien dat er aanzienlijke verbeteringen te realiseren zijn tov. bestaande communicatieprotocollen door op een laag niveau in te grijpen en zo snel als mogelijk is te reageren op kwaliteitsveranderingen in de draadloze verbinding. Deze veranderingen kunnen veroorzaakt zijn door allerlei factoren zoals verplaatsing van de zender of ontvanger en interferentie door andere apparatuur of obstakels. Ons hybride *link-adaptation* algoritme past de ruwe transmissie snelheid (modulatie schema) mbv. van twee methodes aan. Ten eerste

wordt er een standaard rate-controller gebruikt die de kwaliteit op de lange duur optimaliseert mbv. statistieken over de geleverde prestaties. Deze klassieke feedback methode wordt gecomplementeerd met een tweede controller die de momentane condities van de verbinding gebruikt om de gekozen snelheden te beperken tot diegene die de meeste kans maken correct gedemoduleerd te worden door de ontvanger. (Geavanceerde modulatie technieken leveren hoge datasnelheden op maar vereisen goede condities in vergelijking met eenvoudige technieken die robuuster zijn). De resultaten in hoofdstuk 3 laten zien dat ons *link-adaptation* algoritme tot een drastische vermindering van verloren datapakketten leidt in extreme situaties.

Om de prestaties van streaming applicaties verder te verbeteren hebben we mogelijk gemaakt dat het link-adaptation algoritme de status informatie over de draadloze verbinding doorgeeft aan de applicatie zelf. Deze *cross-layer* aanpak staat applicaties toe zich (langzaam) aan te passen aan bijvoorbeeld de effectieve datasnelheid over de verbinding. In hoofdstuk 4 laten we zien dat het aanpassen van de compressiefactor die een video codec gebruikt om beelden over te zenden een aanzienlijke verbetering oplevert voor de menselijke perceptie van de ontvangen uitzending over een draadloze verbinding waarvan de kwaliteit zodanig vermindert dat deze niet door link-adaptation alleen opgevangen kan worden. Hoofdstuk 5 bevat ons voorstel voor een generieke cross-layer interface tussen link-adaptation en applicatie. Deze is zodanig opgezet dat die naadloos geïntegreerd kan worden met bestaande interfaces voor draadloze systemen zoals de veelgebruikte 802.11 standaard.

Ivaylo Haratcherev

Curriculum Vitae

Ivaylo Haratcherev was born in Sofia, Bulgaria on February 9, 1975. He received his M.Sc. degree in Systems and Control Engineering from the Technical University of Sofia, Bulgaria in 2000. He became a Ph.D. student in the Parallel and Distributed Systems group at Delft University of Technology in 2001. His research interests include automatic parameter control and QoS in wireless communications, embedded systems, and real-time control systems.

He is currently employed as a PostDoc in the Parallel and Distributed Systems group at Delft University of Technology.

Publications

1. I. Haratcherev, K. Langendoen, R. Lagendijk and H. Sips. Optimized Video-Streaming over 802.11 by Cross-Layer Signaling. *IEEE Communications Magazine*, special issue on Cross-Layer Protocol Engineering For Wireless Mobile Networks (Part II), Vol. 44(1), January 2006.

2. I. Haratcherev, K. Langendoen, R. Lagendijk and H. Sips. Link Adaptation and Cross-Layer Signaling for Wireless Video-Streaming in a Shared Medium. *IEEE International Conference on Wireless Networks, Communication and Mobile Computing (WirelessCom)*, Maui, Hawaii, USA, June 2005.

3. I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips. Fast 802.11 link adaptation for real-time video streaming by cross-layer signaling. *IEEE Symposium on Circuits and Systems (ISCAS)*, Kobe, Japan, May 2005.

4. I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk and H. Sips. Automatic IEEE 802.11 Rate Control for Streaming Applications. *Wireless Communications and Mobile Computing (Wiley Interscience)*, special issue on Radio Link and Transport Protocol Engineering for Future-Generation Wireless Mobile Data Networks, Vol. 5(4), June 2005.

5. I. Haratcherev, K. Langendoen, R. Lagendijk and H. Sips. Hybrid Rate Control for IEEE 802.11. *ACM International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, Philadelphia, PA, October 2004. (Best paper award)

6. J. Taal, I. Haratcherev, K. Langendoen and R. Lagendijk. Quality of Service Controlled Adaptive Video Coding over IEEE 802.11 Wireless Links. *IEEE International Conference on Multimedia & Expo (ICME)*, special session on Networked Video, Baltimore, MD, July 2003.