INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

No attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

THÈSE

pour obtenir le grade de

DOCTEUR DE L'INPG

Spécialité : « Informatique : Systèmes et Communications »

préparée au laboratoire LSR – IMAG
dans le cadre de l'École Doctorale
« Mathématiques, Sciences et Technologies de l'Information »

présentée et soutenue publiquement par

Paul STARZETZ

MERCREDI 14 JUIN, 2006

Amélioration des performances de bout-en-bout dans des réseaux sans fil

Directeur de thèse: M. Andrzej DUDA

Co-directeur de thèse: Franck ROUSSEAU

JURY:

M. Jacques CHASSIN DE KERGOMMEAUX,	Président
Mme. Pascale PRIMET VICAT-BLANC,	Rapporteur
M. Laurent TOUTAIN,	Rapporteur
M. Andrzej DUDA,	Directeur de thèse
M. Franck ROUSSEAU,	Co-directeur de thèse

Contents

1 Introduction **11**

1.1 Motivation . . . 11
1.2 Goal of the Thesis . . . 11
1.3 Context of the Thesis . . . 12
1.4 Organisation of the Manuscript . . . 13

2 Summary Contribution **15**

2.1 Methodology . . . 15
2.2 Wireless Channel Access Methods . . . 16
2.3 End-to-End Performance in WLANs . . . 19
2.4 Evolution of TCP . . . 20

I State of the Art **21**

3 Wireless Local Area Networks **23**

3.1 History of IEEE 802.11 . . . 23
3.2 IEEE 802.11 Standard and Architecture . . . 25
3.3 Description of IEEE 802.11 Layers . . . 27
3.3.1 802.11 PHY Layer . . . 27
3.3.2 802.11 MAC Layer . . . 29
3.3.3 Overview of 802.11 Frame Types . . . 40
3.4 Related Wireless Standards . . . 43
3.4.1 HiperLan and HiperLan2 . . . 43
3.4.2 HomeRF's SWAP . . . 45
3.4.3 Bluetooth . . . 45
3.5 Conclusion . . . 47

4 TCP : Transmission Control Protocol **49**

4.1 Key Features of TCP . . . 49
4.2 Basic TCP Operation . . . 50
4.2.1 Connection Establishment and Tear Down . . . 50
4.2.2 Sliding Window . . . 51
4.2.3 Acknowledgements and Retransmissions . . . 52
4.3 TCP Packet Format . . . 53
4.4 TCP Congestion Control and Avoidance . . . 55
4.4.1 Slow Start . . . 56
4.4.2 Congestion Avoidance . . . 57
4.4.3 Fast Retransmit . . . 57
4.4.4 Fast Recovery . . . 58
4.5 Next Generation TCP Algorithms . . . 58
4.5.1 TCP Vegas . . . 59
4.5.2 TCP Westwood . . . 62
4.5.3 ECN-TCP : Explicit Congestion Notification . . . 63
4.6 Other Next-Generation Protocols . . . 65

4.6.1 SCTP : Stream Control Transport Protocol 65
4.6.2 UDT : UDP Based Data Transfer 66
4.6.3 DCCP : Datagram Congestion Control Protocol 67
4.7 Conclusion 67

5 TCP in Wireless Environments 69
5.1 Introduction 69
5.2 TCP over Lossy Links 70
5.2.1 I-TCP : Split Connection Approach 70
5.2.2 Snoop-TCP : a Link Layer Solution 71
5.2.3 SACK-TCP : Selective Acknowledgements 72
5.2.4 TCP Decoupling : TCP Connection Shielding 73
5.3 Conclusion 73

II Analysis of Performance Problems in WLANs 75

6 Evaluation of 802.11 MAC Performance 77
6.1 Chapter's goal 77
6.2 Hardware Setup Description 77
6.3 Benchmark Software Description 78
6.3.1 CBR : Constant Bit Rate Source for Linux 78
6.3.2 HTBench : a Concurrent HTTP Benchmark 80
6.4 Experimental Results 81
6.4.1 802.11 DCF Conformance Test 81
6.4.2 CSMA/CA Equal Access Probability Test 84
6.4.3 Throughput Tests 86
6.5 Conclusion 86

7 Problem of TCP Asymmetry and Fairness in WLAN 89
7.1 Wireless Access Scenarios and TCP Behaviour 89
7.1.1 Simple Download Case : Verification in NS 91
7.1.2 Mixed Upload-Download Scenario 100
7.2 Proposed Solutions for Fairness in WLANs 102
7.2.1 DFS : Distributed Fair Scheduling 102
7.2.2 ELF : Effort Limited Fair Queueing 104
7.3 Conclusion 105

III Contributions 107

8 Statistical Analysis of 802.11 and its Improvement 109
8.1 Hash-based CSMA/CA 109
8.1.1 Simplified H-table approach 110
8.1.2 Improved H-table Scheme 113
8.2 Simulation Results 119
8.3 Conclusion and Future Research 130

9 Improving End-to-End Performance of 802.11 WLANs 135
9.1 Motivation and Problem Description 135
9.2 Virtual Flow Queueing for TCP 136
9.2.1 Generalised WFQ Queueing 136
9.2.2 Application to Wireless Networks : VFQ 138
9.3 Validation and Prototype 144
9.3.1 Validation in the NS2 Simulator 144
9.3.2 Linux Experimentation 150

9.4 Other Solutions for the TCP Asymmetry Problem 157
9.5 Discussion and Future Research 158

10 Towards Improving TCP Congestion Control 163
10.1 Introduction and Problem Statement 163
10.2 Presentation of TCP-D 165
10.3 Implementation and Validation in NS2 170
10.4 Conclusion and Future Work 175

11 Conclusion and Perspectives 179
11.1 Wireless Channel Access Methods 179
11.1.1 Achievements 179
11.1.2 Perspectives 182
11.2 End-to-End Performance in WLANs 184
11.2.1 Achievements 184
11.2.2 Perspectives 184
11.3 Evolution of TCP 187
11.3.1 Achievements 187
11.3.2 Perspectives 187

List of Figures

3.1 The IEEE 802 network protocol family. 23
3.2 The IEEE 802.11 wireless network family. 25
3.3 A typical IEEE 802.11 wireless network installation 26
3.4 Frequency bands available for 802.11(b) WLANs 27
3.5 Basic CSMA/CA channel access procedure. 29
3.6 IEEE 802.11 virtual carrier sense using RTS/CTS (request- and clear-to-send). . . 31
3.7 Subsequent 802.11 frame transfers with backoff. 32
3.8 Timing of 802.11 conversation with MAC level acknowledgement. 34
3.9 Complete 802.11 transaction with virtual carrier sense and frame fragmentation. . 35
3.10 Coexistence of DCF and PCF in the 802.11 superframe. 38
3.11 Structure of the physically transmitted 802.11 frame. The variable length MAC Protocol Data Unit (MPDU) contains the LL (Link Layer) frame. 40
3.12 Format of the 802.11 LL frame. 41
3.13 Structure of the frame control field of the 802.11 LL frame. 41

4.1 The TCP sliding window principle. 51
4.2 TCP header format. 54

6.1 Experimental 802.11 conformance test #1 : inter-packet spaces for a maximum speed UDP source. 82
6.2 The same data as in figure 6.1 shown in a single column. 83
6.3 IEEE 802.11 conformance test #2 : inter-packet spacing for a 32kb/s CBR UDP source. 84
6.4 802.11 CSMA equal channel access probability test. 85

7.1 Wireless network topology used in NS2 simulations 91
7.2 Scenario 1 : simultaneous TCP downloads, 6 NewReno wireless stations, transferred volume shown. 92
7.3 Scenario 1 : Jain's fairness index computed with a resolution of $\delta = 1s$. 93
7.4 Wireless scenario 1 : queue occupancy for AP and mobiles. Upper plot shows a zoom into the lower plot. 95
7.5 Wireless scenario 1 : congestion window size evolution with local synchronisation (upper plot zoomed). 96
7.6 Scenario 2 : Vegas(2,4) and NewReno transfer volume and fairness index, TCP download 98
7.7 Simulation scenario 3 : three stations with one TCP download (low bandwidth) and three with one upload (high bandwidth) under DropTail at the AP. 100
7.8 Composite Jain's fairness index for scenario 3 (3 upload + 3 download stations, resolution 1s). 101

8.1 Approximated optimal PDF for $T(s) = 4$ in the simplified 2 nodes case. 117
8.2 Effective contention window sizes for 802.11, Idle Sense and HASH1 access methods. 120
8.3 Simulated collision rates for 802.11, Idle Sense and HASH1 channel access method. 121
8.4 Average simulated transmission wait for 802.11, Idle Sense and HASH1. 123
8.5 Average simulated collision wait for 802.11, Idle Sense and HASH1. 124

8.6 Jain's fairness index for Idle Sense (solid symbols) and HASH1. 125
8.7 Self-correlation index for HASH1, 802.11 and Idle Sense, 5 contending stations. . . 126
8.8 Self-correlation index for HASH1, 802.11 and Idle Sense, 15 contending nodes. . . . 127
8.9 Jain's fairness index for 802.11 and the modified version 802.11-M. 128
8.10 Effective contention window size of 802.11-M. 129
8.11 Self-correlation index for 802.11 and modified 802.11 for 15 contending nodes. . . . 130
8.12 The SBB and SAB scheduling policies. 132

9.1 Example 802.11 network used as Internet or company access gateway for mobile nodes. 139
9.2 The coarse architecture of the VF^2Q scheduler. 140
9.3 Architecture of the TCP node "black box". Two adjoint virtual flow descriptor pairs are shown. 141
9.4 The WFQ cost function versus the VFQ cost function for a 802.11b wireless network. 142
9.5 Scenario 3 under VFQ queueing discipline, 3 download and 3 upload flows obtain perfectly equal bandwidth (the 6 curves overlap). 146
9.6 Jain's fairness index for 3 downloading and 3 uploading TCP wireless nodes under VFQ queueing. 147
9.7 Comparison of the smoothened TCP RTT estimates for upload (dotted lines) and download direction. Upper graph : 3+3 nodes under drop tail, lower graph same scenario under VFQ. 148
9.8 Scenario 3, congestion window development under the VFQ discipline. 149
9.9 Scenario 3, VFQ with weights set to 2 for the down- and to 1 for the upload direction. 150
9.10 Simulation with two distinct web servers : drop tail at the AP, 3 Newreno flows with link delay 10ms, 3 flows with 100ms 151
9.11 Two distinct web servers, VFQ, 3x Newreno at 10ms, 3x at 100ms 152
9.12 Two distinct web servers, VFQ, 3x Newreno at 10ms, 3x at 50ms 153
9.13 Linux TCP transfers over 802.11b, Drop Tail at the AP, queue length set to 30 packets. 154
9.14 Linux TCP transfers over 802.11b, 3 downloads and one upload, DropTail q=40 packets. 155
9.15 Linux TCP transfers over 802.11b, 3 downloads (curves overlapping) and one upload, simple WFQ, queue length=10 pkt. per class. 156
9.16 Linux TCP transfers over 802.11b, 3 downloads and one upload, VFQ with queue length=10 pkt. per station. 157

10.1 Typical TCP goodput curve over the normalised congestion window (W=1 : congestion window size equal to bandwidth-delay product). 165
10.2 Generalised TCP window probing : Reno-like sawtooth (solid curve) and a perfect harmonic oscillation (dashed curve) 167
10.3 Scenario 1 : six TCP-D downloads, transferred volume (upper plot) and the Jain's fairness index (lower plot) shown. 171
10.4 Sender-side congestion window size development for scenario 1, upper graph shows a zoom at the cold-start phase. 172
10.5 Queue length development in scenario 1 at the access point. 173
10.6 Five TCPD flows in direct competition with one Reno flow 174
10.7 Five Reno flows in direct competition with one TCPD flow 175
10.8 Five TCP Vegas flows in direct competition with one TCPD flow 176

List of Tables

3.1 Actual IEEE 802.11 WLAN standards 25
3.2 Today's Wireless Technologies 47

6.1 Average WLAN throughput 86

7.1 NS2 802.11 module MAC settings used in simulation. 91
7.2 TCP Algorithms Summary 105

9.1 VFQ TCL configuration parameters in NS 145

Chapter 1

Introduction

1.1 Motivation

The rapid and unexpected growth of the Internet and related technologies into the commercial world as well as into private homes in the recent ten years resulted in highly increasing research interest in networks, especially those based on the IP (Internet Protocol). While the original design goal of the TCP/IP protocol suite was to provide a robust mechanism for constructing a distributed network infrastructure insensitive to damages and arose from various defence research projects, such huge adaptiveness to unpredictable and unforeseen changes turned out to be its biggest advantage in today's world of rapidly evolving networking technologies.

Among all the numerous new technologies which have been incorporated into IP networks in recent past are also wireless and mobile technologies permitting the IP technology to spread into places, where it could not be used before due to e.g. physical constraints. While the winner in the race between the concurrent wireless technologies has not been determined yet, it slowly becomes apparent that the IEEE 802.11 WLAN and succeeding wireless standards will be playing a key role in the future of an ubiquitous Internet. Wireless links based on this relatively simple and cheap technology are not without problems, though as they are very sensitive to interference, signal quality decreases as a mobile moves away from its access point and transmission speeds are not always as high as those offered by actual wireline LANs.

While the effort spent on those emerging WLAN technologies was mostly devoted to providing seamless integration into existing IP architectures, many questions are still open, for example in the case that such WLAN technology is going to replace wireline links at the network edge. More specifically, how does actual WLAN technology like IEEE 802.11 integrate with state-of-the-art wireline core networks and are there any unanswered performance issues, if those oldish but wide-spread wireline-IP protocols like TCP are going to be used in a heterogeneous network with WLAN links?

1.2 Goal of the Thesis

In this work we try to identify, to analyse and to answer those performance questions, which did not receive significant and sufficient attention in the research performed during last years. A central role for this thesis plays the problem of fairness, that is a fair (equal) provisioning of available resources in the network (e.g. throughput) to multiple participants resulting in a fair performance experience (e.g. delay, packet loss) by the user. While classical QoS research concentrates rather on flow (or aggregate) scheduling at the packet level and resulting issues, we try to investigate the end-to-end performance aspects in the context of WLAN/LAN integration scenarios.

Fairness has not been considered a fundamental QoS issue for years, however, recently this subject is getting increased attention in the research community. Thus in this thesis we focus on those less explored aspects of WLAN technologies by investigating the performance, namely the throughput, the delay and the fairness achievable in state-of-the-art IEEE 802.11 WLANs. Even though the 802.11 standard is quickly evolving and implementations are maturing, the very basic building blocks of 802.11 like the MAC access method did not change too much since the initial version of the standard. Therefore it is interesting but also very challenging to try to improve such an "invariant" constituent part of the standard.

Moreover, since networking protocols of a particular OSI layer (e.g. like Ethernet CSMA/CD) are rarely designed with upper or lower layers in mind (e.g. taking into account TCP traffic patterns as enforced by modern media applications), it is interesting and important to investigate the interplay of a novel protocol like IEEE 802.11 with established ones (e.g. like deployed TCP). Experience teaches that such "out-of-the-box" solutions do not necessarily provide satisfactory performance, if no special care is taken about correctly tuning and adapting the different layers to each other. An example could be for instance a machine installed with an out-of-the-box Linux system but connected to a slow dialup connection, where the default interface queue size of the operating system is far too big for the speed of the dial-up connection, resulting in very bad overall performance experience. Similar negative interactions may be the result, if novel technologies like 802.11 are not carefully used thus entailing dissatisfied users and diminishing the acceptance of the WLAN technology.

1.3 Context of the Thesis

This work was carried out at the LSR laboratory in Grenoble. It must also be mentioned that this thesis begun with a participation in the @IRS Project [70], a French national project for next-generation networks, which was the initial context of this work. But since the project was already in an end-stage, it was difficult to contribute very novel results other than already presented in the thesis [91]. Thus the work performed for the @IRS project only consisted of a clean reimplementation and evaluation of the @IRS architecture (developed so far for BSD) under the Linux operating system.

A further investigation of the @IRS QoS framework permitted to design a QoS module for the Apache web server [9], [8], which was then implemented during the course of a student project and evaluated for the Linux @IRS DiffServ QoS architecture in a IEEE 802.11b WLAN environment. But since the participation in the @IRS project begun in its final stage just right before its completion, further research of this thesis took a different path. Therefore this work investigates questions more at the base of the WLAN/LAN integration, slightly apart from architecture-only issues.

1.4 Organisation of the Manuscript

The complexity of networking technologies requires that the investigation of any performance problems must be broken down into simpler pieces, usually at boundaries marking different OSI layers (e.g. MAC, IP, etc.) or constituents related to the physical or logical composition of the network infrastructure (e.g. end-host, edge router, core router). Therefore the following manuscript is organised into three main parts, not counting the following short 2nd chapter summarising the methodology and the contribution done in each of the sub-areas, namely at the MAC, the IP and the TCP level.

The next four chapters (chapters 3, 4, 5 and 6 in Part I) provide an introduction into the state-of-the-art in the research areas with importance for this work. Chapter 3 introduces and describes popular wireless technologies with increased attention on the IEEE 802.11 WLAN standard and its channel access protocol. Chapter 4 recapitulates fundamental features and functioning principles of the TCP protocol suite and discusses the state-of-the-art in the area of congestion control and avoidance. This is accomplished by describing actual efforts to improve classical TCP upon its congestion avoidance behaviour as well as upon its performance in the context of wireless applications. Finally, chapter 5 sumarises more specialised work done in the area of TCP over high-error rate links using wireless technology.

The second part of the thesis (chapters 6 and 7) provides a phenomenological evaluation of the performance of current 802.11 equipment as well as an evaluation of the behaviour of the 802.11 network under simple scenarios expected to arise, where 802.11 WiFi is used to provide seamless Internet connectivity: at home or work office. Chapter 6 evaluates the existing 802.11 hardware equipment with respect to correct implementation of the 802.11 standard and shows that theoretically 802.11 conformant devices (that is "WiFi certified") can exhibit different performance in practice. Then chapter 7 analyses the behaviour of an 802.11 wireless network in a simple topology and with respect to the perceived performance in scenarios, which are expected to appear in the daily use of WiFi equipment. It identifies practical performance issues in IEEE 802.11 requiring special support from the wireless network infrastructure (active queue), in order to provide acceptable end-to-end performance for classical TCP protocols.

The third part of this thesis (chapters 8, 9, 10, 11) presents the main contributions of this work. This part is built on the previous introductory chapters and makes an attempt to define but also to fill the gaps not receiving sufficient attention in recent literature in order to finally synthesise a generalised solution for the identified core problem. In chapter 8 a statistical approach to describe and quantify the CSMA/CA class of MAC algorithms is presented. The 8th chapter delivers novel insights into the structure of the IEEE CSMA/CA channel access algorithm but also into the generalised problem of collision avoidance on a shared channel. The classical Jain's fairness index widely used to quantify short-term fairness in multi-access protocols is complemented by another statistical metrics: the channel autocorrelation function. By application of this statistical measure the real source of short-term fairness or unfairness in those class of algorithms is identified. In chapter 9 a generalisation of fair queueing algorithms is described and used to improve 802.11 fairness, which may be unsatisfactory for the wireless network acting as an Internet access gateway. The developed active queue management solution for TCP over wireless called VFQ is then evaluated in the NS2 network simulator but also implemented for and tested under the Linux kernel network stack. Chapter 10 goes a step further to look at the TCP layer. The chapter is an attempt to propose an improvement (in the sense of short-time fairness and in the sense of improved average congestion avoidance) to the congestion control algorithm of the classical (Reno-like) TCP protocol. The improvement called TCP-D is based on the observation (through NS2 simulations) of the behaviour of TCP congestion control in simplified WLAN/LAN integration scenarios, where very few wireless clients contribute the major part to the bottleneck's congestion level. Finally, Chapter 11 sumarises and concludes the results from each sub-contribution.

Chapter 2

Summary Contribution

2.1 Methodology

In this work we use a dual methodology to derive the results in each sub-contribution. At the beginning of the process there is always an observation of a novel phenomenon, which requires further analysis and an explanation. Then a reduction of the observed system behaviour is performed to recognise the core of the problem and to find a specialised solution. However on the other hand the specialised solution is shown to be just an application of a more general idea which is found by intuition and by analogy to other even natural systems. The goal of this thesis is thus not only to provide a working technical solution each time, but also (in each sub-contribution) to establish relationships between things believed previously to be unrelated or without any previous evidence for equivalence.

Why it is so difficult to analyse networking systems and why the results are sometimes so controversial and even contradictory? The problem seems to lie in the growing complexity of the technologies used at each networking layer. While it is surely possible to design each individual component of a networking system to respect certain performance bounds for a given traffic pattern, the interaction of such multiple building blocks may not necessarily yield a performant system in practice. A simple and well known example of such complexity interaction is the behaviour of one and many TCP flows, where the behaviour of only one flow can be rather easily quantified for well defined network parameters, while on the other hand a system of many TCP flows can show very different operation regimes difficult to quantify by mathematical means [111].

An additional difficulty arises in areas, where the research is going very fast and standards are rather drafts in the conventional meaning of this word at the moment of their public release. A good example therefore is the IEEE 802.11 wireless standard, which is a primary subject of this thesis. Its complexity is the source of numerous performance issues ([58], [59], [21], [81], [136], [115]), while being still subject to various proposed modifications and improvements ([60], [100], [134], [18], [88]) even more than ten years after the approval of the initial version of the standard. On the other hand the basic access method in 802.11 called DCF has been found to be very difficult subject for exact mathematical modelling ([22], [139]), thus requiring and necessitating the (re)search for simplified models and algorithms. Moreover, due to the high complexity of upper layers involved in the transmission of user data, e.g. TCP segmentation, application traffic patterns and OS kernel behaviour (and so on), the compound system where wireless technology is being used can be expected to behave rather unpredictably.

Thus an experimental methodology based primarily on the observation of phenomena remains a valid scientific approach in network research. However the process must be objective so that the researcher does not bias the interpretation of the results or change the results outright. Therefore it is required, if even not obligatory that results gained in practical experimentation should be also validated in simulation (if possible and applicable to the given problem) permitting so to neglect details of a particular (maybe even faulty) implementation. Another basic expectation is that of making documentation of data and methodology available for careful validation by other scientists, thereby allowing other researchers the opportunity to verify the results. The experimental method should also involve an attempt to achieve control over the key factors ("control knobs") of the investigated system, which may in turn be manipulated to test new hypotheses in order to gain further and novel knowledge.

2.2 Wireless Channel Access Methods

The contribution presented in this manuscript is to answer the question about the structure of an optimal collision avoidance algorithm for wireless networks. This question has been investigated in the context of two recent and interesting proposals improving the basic 802.11 access scheme, namely Idle Sense [60] and TCF [87]. The initial observation used for in this part of my work is the intuitive finding that there is a relation between classical hashing problems and collision avoidance. Further I use the observation that not only the exponential backoff algorithm of 802.11 (see 802.11 chapter for details) influences the MAC level, short-time fairness, but also the residual backoff procedure (which has not been noticed in relevant literature), thus permitting the assumption that there should be a more general principle governing the short-time fairness of a channel access scheme.

My work provides an alternative view at the CSMA/CA type access schemes and relates these group of algorithms to distributed hashing and sorting. It is known that the DCF method in IEEE 802.11 is very difficult to analyse by mathematical means as shown in the articles [22] and [139]. The simplified model for CA presented here shows that it is possible to construct far less complex collision avoidance algorithms without a significant (in the sense of being classified as already unfair with respect to established literature) sacrifice of short-term fairness, which is believed to be a fundamental issue in those class of algorithms [59] [21] [83].

In this chapter a reference model called $\mathcal{H}$ suitable for performing comparisons of different collision avoidance schemes has been constructed. It uses an uniform random distribution of backoff slots over a fixed size hash table of C places. For each reasonable (that is relevant in practice) table size C and increasing number N of contending stations the $\mathcal{H}$-scheme has been simulated and the average collision probability $p_c(C, N)$ as well as the average number of empty slots (wait time) before a successful transmission $W_T(C, N)$ and before a (visible) collision $W_C(C, N)$ has been calculated.

If another collision avoidance algorithm (like standard IEEE 802.11) is then given, its statistical properties can be quantified by comparing its average $\bar{W_T}(N)$, $\bar{W_C}(N)$ and $\bar{p_c}(N)$ values (obtained in a respective simulation of that algorithm) with values from the simple hash table scheme at the point, where $\bar{p_c}(N) = p_c(C, N)$ for a fixed number of contending stations N. Simultaneously an effective statistical window size $\bar{C}$ of the algorithm in question can be defined and obtained by taking the respective C value of the $\mathcal{H}$-scheme for equal collision rate.

While the simple[1] $\mathcal{H}$ access scheme is just a construction well suitable for building a reference system, a practical channel access algorithm must solve a trade-off problem between on the one hand as low as possible visible (that is appearing on the channel) collision rate and on the other hand low packet wait before each transmission respective collision event. This requirement means that in a probabilistic system[2] the tradeoff to be solved is the time wasted in channel collisions plus the time wasted in waiting due to idle slots before each transmission attempt. This tradeoff problem can be quantified for any arbitrary probability distribution $p(t)$ by introducing a cost functional of $p(t)$ and its respective CDF ($c(t)$ cumulative probability distribution).

To solve the tradeoff problem the Euler-Lagrange differential equation is applied to a simplified version of the optimisation task in order to find an improved probability distribution of the $\mathcal{H}$-system. The solution delivers insights about the structure of the optimal probability distribution for $\mathcal{H}$, however the exact solution of the problem could not be found due to its mathematical complexity. Despite this difficulty the insight found in this part of my work is that in order to only avoid channel collisions it is optimal to spread the stations over all of the existing C table slots with equal probability $p = \frac{1}{C}$. On the other hand in order to optimise the total cost with respect to the utilisation of $\mathcal{H}$ as a wireless access algorithm, it is more advantageous to use slots at the beginning of the table to decrease the average packet wait delay. The trade-off between the two contrary goals yields a shape of the probability distribution close to a straight line with a negative slope.

Another contribution of this part of the work is to provide another, complementary statistical quantification for short-term fairness. The short-term fairness defined by any $\mathcal{H}$ (that is for an arbitrary $p(t)$) system has been found to provide a reference point for defining short-term fairness of other collision avoidance schemes. By intuition another statistical metrics different from the widely used Jain's fairness index J [74] has been introduced. It quantifies the deviation in the short-time fairness of an arbitrary access algorithm $\mathcal{A}$ from an equal cycle-by-cycle probability scheme like $\mathcal{H}$ and is called the self-correlation probability p_s. While the classical fairness index J gives a "fairness score" to a CA scheme for a particular time window, p_s provides (along with the reference value of p_s in $\mathcal{H}$) an insight into why a particular CA scheme is less or more fair for that time window by quantifying the correlation of the CA scheme.

[1]**constant** $p(t)$.

[2]**no "hidden variables" in the stations,** $p(t)$ **not time-dependent.**

The self-correlation probability index $p_s(t)$ is defined in the context of the wireless channel access problem as the probability that a station, which has sent a packet at the time T will have sent another packet at the time $T+t$, averaged over a sequence of channel access events. In practice it makes more sense to define $p_s(t)$ not over time but over a sequence of (successful) wireless transmission cycles marking each of the N nodes with an unique ID $n_i, i = 1 \dots N$.

To show the validity of the approach using the novel p_s index, an ad-hoc modifications to the original 802.11 DCF is investigated. The result supports the initial hypothesis, that the short-term fairness problem in wireless networks (and in other shared channel access algorithms) is not directly related to the structure of the backoff algorithm executed in case of collisions, neither to the modifications to the contention window size performed on each retransmission in case of 802.11. The relation between fairness and the backoff procedure is only indirect: by the correlation between the transmission probabilities for a given node at a reference time T and a later time $T+t$. Therefore the real source of short-term unfairness has been identified as the positive time-correlation between consecutive packet transmissions of a given station and a quantification metrics ("control knob") measured by the autocorrelation coefficient $p_s(t)$ of the CA scheme in question has been presented.

Further achievement of this work is to formalise the problem of collision avoidance and relate this group of algorithms to distributed hashing and sorting. While a sorting algorithm requires that a (partial) ordering operator "<=" is defined, generalised collision avoidance schemes may be defined as a group of algorithms using a distributed version of the sorting operator called $\mathcal{M}$. It can be defined as the mapping $\mathcal{M} : \mathcal{S} \rightarrow C$ from the set $\mathcal{S}$ of all $S_j \in \mathcal{S}$ of the backoff timer values of each backlogged station onto the image $C = \{0, 1\}$ and defined for non-empty S_j as:

$$m_j = \begin{cases} 0 & \text{if } b_j^{i_k} = min(S_j) \wedge \forall l \neq k \;\; b_j^{i_k} \neq b_j^{i_l} \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

where m_j stands for the value of $\mathcal{M}$ in the contention cycle j and means $m = 0$ for successful transmission event and $m = 1$ for channel collision. Note that $m_j = 0$ means also for the contention winner numbered with l (that is the station which computed $m_j = 0$) that $b_j^{i_l} < b_j^{i_k} \forall k$ and this is the rationale for naming $\mathcal{M}$ the distributed ordering operator. Therefore $\mathcal{M}$ can be understood as an elementary coupling (information exchange) between contending stations and their backoff timers. Its value can be computed locally (locality) by any participating station in each contention cycle.

This formal definition of $\mathcal{M}$ allows to design a broad spectrum of distributed and cooperative channel algorithms based solely on the locally computable value of m_j, for example such minimising the expectation value $< m_j >$ (that is minimising the average collision rates in equilibrium) for $j \rightarrow \infty$. For a noisy channel the expression for m_j must by augmented by a discrete random variate (or a variate resembling the error behaviour of the wireless channel e.g. bursts) $\sigma_j : j \rightarrow \{0, 1\}$ with an expectation value $< \sigma >$ between 0 (no wireless errors) and 1 (100% loss). Depending on how the value of m_j is used in each contention cycle (whether the information contained in m_j is honored or not) a gracefull transition between a temporal sorted system (distributed sorting) and a probabilistic system (distributed hashing) is possible.

2.3 End-to-End Performance in WLANs

The contribution of this part of my work is to shows that just an one-layer solution like packet level scheduling at the access point in the downstream direction is not able to sufficiently solve the fairness as well as the bandwidth provisioning problem in an dynamic WLAN environment. This issue is related to the characteristics of the physical packet transmission based on the 802.11 DCF as well as to the expected average utilisation scenario of the network resources in places, where WLANs are used to provide seamless Internet connectivity.

The experimentation as well as simulation results presented in my work reveal potential fairness problems (upload-download asymmetry), if the WLAN traffic is dominated by classical TCP sources. The core of the problem is identified as different sensitivity of TCP sessions to packet drops at the access point and is prevalent if the access point has no priority access to the wireless channel while servicing TCP traffic for multiple wireless end-stations. It is important to note at this place that a pure analytic approach regarding just one layer each time would be unable to catch the asymmetry problem, since it is not directly related to the behaviour of one networking layer but rather to the interaction between different layers designed in theorey to behave well (e.g. 802.11 MAC is fine in this sense for stations using best-effort UDP).

Second achievement in this research area is the extension of the WFQ (weighted fair queueing) family of algorithms to imply a broader notion of network ressource utilisation by relating the virtual service times of QoS classes to physical observables through the introduction of a suitable cost mapping function. This modification and generalisation of WFQ permits to implement a variety of different queueing disciplines providing fairness not only directly at the observed throughput level but at the level of a different physical quantity (called "observable", e.g. to implement weighted round robin scheduling by selecting the observable to be the average packet rate).

The extension of WFQ is then applied to the wireless case and the virtual flow queueing (VFQ) scheduler for TCP is presented in this work. VFQ is based on the wireless transmission time cost function and regulates TCP flows either by accounting for the service consumed by the observed data packets (download) or for the service deduced from the ACK stream in the opposite direction (upload). It provides a very natural way of scheduling TCP flows in those dynamic WLAN environments with the additional advantage of improving TCP security compared to a two-way type scheduler observing the up- and the downlink direction of a wireless cell directly. While the original goal of VFQ was to restore fairness (due to the upload-download asymmetry) with respect to obtained capacity shares between different WLAN stations on long-term time scale, it is also capable of providing weighted fairness if desired. Another advantage of the virtual flow scheduling approach over other solutions is its easy deployment in the situation, where the uplink and the downlink direction of a virtual link can not be observed at a single physical node.

2.4 Evolution of TCP

The contribution done in this part of my work is the observation and analysis of the behaviour of TCP in wireless scenarios relevant in practice. Moreover the investigation of these simple scenarios leads to the conclusion, that classical TCP is a deficitary algorithm with respect to the short-time fairness as well as to long-term fairness under certain circumstances.

The intended congestion avoidance of widely deployed TCP algorithms does not always successfully avoid network congestion, e.g. in a few-users scenario as imposed by a wireless-wired integration situation. The core of the short-term fairness problem is the aggressive window probing of Reno-like TCP, which must overflow the bottleneck link queue first, in order to estimate the available capacity along its path. Improvements of the widespread Reno-like TCP algorithm like Vegas-TCP as studied through NS simulations don't seem to substantially improve the perceived performance with respect to throughput fairness and congestion avoidance (that is low end-to-end delay) in these scenarios.

Here again the importance of an empirical method becomes visible: while it is possible to derive mathematical models describing the variation of system parameters (bottleneck queue length, congestion window size and derived from them delay and packet loss rate, for example) over time for TCP over simple topologies (e.g. [24]), those models are usually limited to just one aspect of the problem, e.g. to the study of the behaviour of the particular (Reno [71], Vegas [25], Westwood [29], etc.) congestion avoidance scheme itself, thus being frequently unable to quantify the "total outcome" in the sense of the perceived performance for the end user, if the real scenario differs substantially from the simplified model e.g. by presence of a wireless link exhibiting more complex transmission characteristics than an idealised reliable link.

Based on those experimental findings an idea for implementing a differential type of congestion control is developed and investigated in a preliminary TCP-D prototype for the NS2 network simulator. The brief analysis of the TCP-D prototype shows that the algorithm works for simplified wireless scenarios and can further improve the end-to-end performance with respect to short-term but also long-term throughput fairness. Moreover an improved congestion avoidance algorithm like TCP-D can help to alleviate the permanent network congestion level. The key idea of TCP-D is based on an analogy to a harmonic oscillator and deploys a finer-grained control of the congestion window size than in normal Reno-TCP. The rationale for developing TCP-D is based on the observation that classical congestion control algorithms are blind to the absolute level of network congestion and rely solely on the induction of heavy congestion (in absence of RED or other active management schemes), in order to find the right point (or better to say: region in case of Reno) of operation.

Part I

State of the Art

Chapter 3

Wireless Local Area Networks

3.1 History of IEEE 802.11

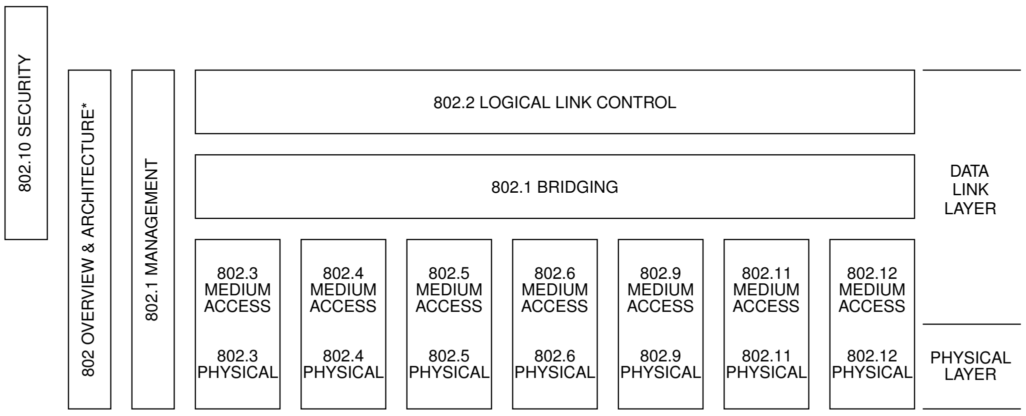

Figure 3.1: The IEEE 802 network protocol family.

The 802.11 standard appeared first in the 1990's developed under the leadership of the Institute of Electrical and Electronics Engineers (IEEE). The initial task of the IEEE's "Project 802.11" committee was to analyse the applications and environments in which wireless networks could be used in future. Such work was urgently necessary since the main problem of radio networks acceptance in the market place at that time was, that there were not just one unique standard like Ethernet with a guaranteed compatibility between all devices, but many proprietary standards pushed by each independent vendor and incompatible between themselves. Because corporate customers require an established unique standard, most of the vendors joined the IEEE in an effort to create a standard for radio LANs.

This joint effort lead to the IEEE 802.11 (like Ethernet is IEEE 802.3, Token Ring is IEEE 802.5 etc.) wireless network architecture standard. Of course, once in the 802.11 committee, each vendor has pushed its own technologies and technical features into the standard in order to try to make the standard closer to his own products. The result is now a standard, which took far too much time for completion, which is overcomplicated and bloated with features and might even become obsolete by newer WLAN technologies before products supporting all of the features come to the market. But it is a standard based on experience, versatile and well designed, including all of the optimisations and clever techniques already developed by the different wireless LAN vendors.

As early as in March 1992 the IEEE committee formally established the functional requirements for a wireless LAN protocol, which has now emerged and expanded to be one of the leading technologies in the wireless world. In order to better understand how it came to the overwhelming expansion of the 802.11 network standard, the most important milestones in the history of 802.11 are described:

- 802.11 (without any additional letter) was the first proposal of the IEEE committee and appeared in early 1992. The 802.11 draft specifies using either FHSS (frequency hopping spread spectrum) or DSSS (direct sequence spread spectrum) and provides 1 and 2 Mbps transmission rates on the Industry, Medical and Scientific (ISM) 2.4GHz band. The 2.4GHz band was chosen because it covers the water dipole's resonance frequencies (same as in a microwave oven). The natural water content in the atmosphere delivers high damping to 2.4GHz microwaves which is required for short range devices like 802.11 wireless LAN. The initial version of the standard was then finalised in 1997.
- 802.11a is one (together with 802.11b) of the two mature proposals of the IEEE committee and appeared two years later after the original 802.11 draft. It is using OFDM (Orthogonal Frequency Division Multiplexing) PHY and operates at 5GHz. This provides a theoretical gross bit rate of up to 54Mbps. Fallback bit rates include 54Mbps, 48Mbps, 36Mbps, 24Mbps, 12Mbps and 6Mbps. The maximum operating range is something about 50m indoors and 100m outdoors[1].
- 802.11b, also known as Wi-Fi or High Rate 802.11 and uses CCK (a variation on M-ary Orthogonal Keying modulation) PHY modulation. It provides a gross data transmission rate of 11Mbps and permits fallbacks to 5.5, 2 and 1 Mbps respectively if required by signal strength. Since the PHY frequency band differs from that one of the 802.11a WLAN, the two wireless network versions are not directly interoperable. In contrast to 802.11a and for the reason of the relatively narrow ISM band size b-channels overlap.
- 802.11g was first proposed by IEEE in November 2001 and ratified in June 2003, however first "pre standard" 802.11g products began shipping already in March 2003. It provides a 54+ Mbps physical transmission rate and runs also on the same 2.4GHz band like 802.11b. Fallback rates equal those for the 802.11a WLAN. Its operating range has been slightly improved compared to 802.11a/b (up to 300m outdoor under ideal operating conditions) due to improved PHY modulation. Recently there is an effort to double the gross data rate of 802.11g products to a total of 108Mbps[2].

This is the actual state of the 802.11 wireless network technology. It is believed that next generation wireless applications will require even higher WLAN

[1]**Without directional antenna.**
[2]**E.g. the Linksys Speedbooster technology.**

data throughput to satisfy high quality multimedia applications and that higher operating range will be demanded[3]. Hence there is a continuous interest in working on next-generation WLAN standards. Actually there are two major groups working on the development of next generation wireless networks: IEEE 802.11n Task Group and the Wi-Fi Alliance.

The objective of the IEEE Task Group working on the 802.11n WLAN standard is to define modifications to the Physical Layer and Media Access Control Layer (PHY/MAC) delivering a minimum usable (that is available at the MAC layer) data rate of 100 Mbps which is believed to require a new PHY providing a physical OTA data rate of circa 200Mbps.

An overview of the IEEE 802.11 WLAN family has been shown in the figure 3.2.

Table 3.1: Actual IEEE 802.11 WLAN standards

802.11 standard	**OTA data rate**	**usable MAC data rate**
802.11a	54 Mbps	25 Mbps
802.11b	11 Mbps	5.5 Mbps
802.11g	54 Mbps	25 Mbps (if 802.11b not present)

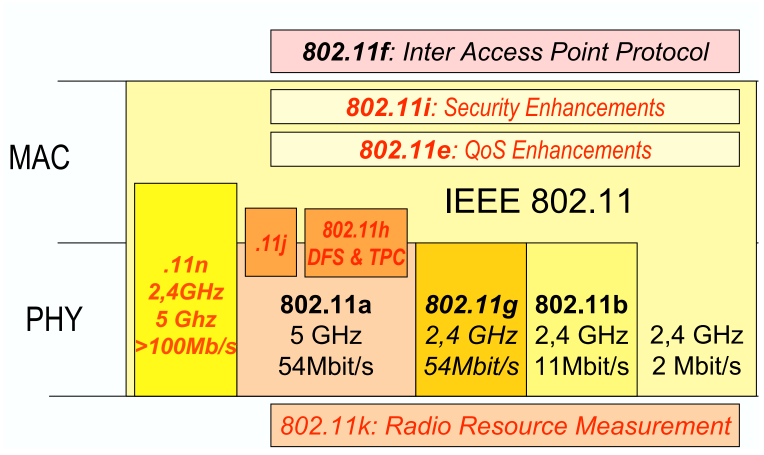
802.11f: Inter Access Point Protocol
802.11i: Security Enhancements
802.11e: QoS Enhancements
MAC
IEEE 802.11
PHY
.11n 2,4GHz 5 Ghz >100Mb/s
.11j
802.11h DFS & TPC
802.11a 5 GHz 54Mbit/s
802.11g 2,4 GHz 54Mbit/s
802.11b 2,4 GHz 11Mbit/s
2,4 GHz 2 Mbit/s
802.11k: Radio Resource Measurement

Figure 3.2: The IEEE 802.11 wireless network family.

3.2 IEEE 802.11 Standard and Architecture

The purpose of this chapter is to give a brief description of the IEEE 802.11 wireless network standard, its basic concepts, the principle of operation and some of the reasons behind the features of 802.11 with focus on the MAC layer and the CSMA/CA access method. The goal of the following however is not to give a very detailed description of all of the 802.11 features, since the standard is extremely bloated. The full 802.11 IEEE WLAN document can be found at the IEEE web page [66] for

[3]There are various ongoing projects e.g. in the USA to provide city-wide WLAN coverage comparable to GSM.

further reading.

An IEEE 802.11 LAN is based on a cellular architecture where the network is subdivided into smaller cells. Each cell (called in the IEEE nomenclature Basic Service Set or BSS) is controlled by a Base Station (called also Access Point and abbreviated from now on with AP). Even though that a wireless LAN may be formed by a single cell with a single access point (and as will be described later, it can even work without any access point device), most production WLAN installations will be formed by several cells, where the access points are connected through some kind of backbone network (called `Distribution System` or DS), typically Ethernet and in rare cases wireless itself.

The whole interconnected Wireless LAN network including the different cells, their

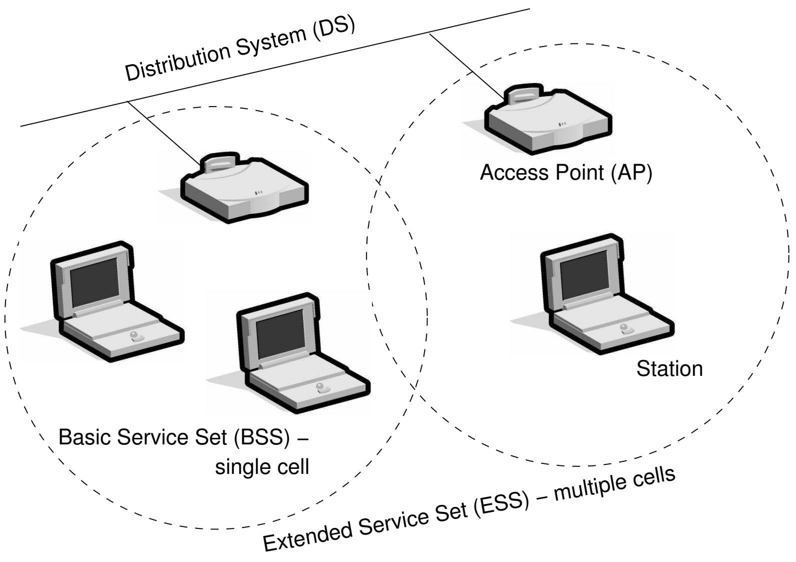
Distribution System (DS)
Access Point (AP)
Station
Basic Service Set (BSS) – single cell
Extended Service Set (ESS) – multiple cells

Figure 3.3: A typical IEEE 802.11 wireless network installation

respective access points and the distribution system appears to the upper layers of the OSI model as a single 802-like network and is called Extended Service Set (ESS). The picture 3.3 shows a typical 802.11 LAN with the components described previously.

Further the IEEE 802.11 standard defines the concept of a `WLAN portal`. A `portal` is a special device which interconnects between a 802.11 and another 802-like LAN. This concept is an abstract description of part of the functionality of a `translation bridge`. A portal can contain additional logic which is necessary to interconnect the two different 802 networks, for example to adapt the relatively slow wireless network to the fast backbone LAN, or for example in case of a company wireless access network, include a `VPN` gateway where wireless stations must authenticate themselves before they are allowed to access other company resources.

Even though the standard does not necessarily request so, typical installations will have the AP and portal on a single physical entity, since the majority of the currently available 802.11 AP products[4] also offer LAN routing interfaces and sometimes already portal functionality (NAT, VPN, Proxy, etc.). It is also worth noting that there is a huge amount of open source software for building 802.11 access points and portals based on the Linux operating system and consumer 802.11 interface cards. I call this type of a combined access point and portal a `Software Access Point`. Using open source software facilitate the task of the network administrator if upgrade or replacement of WLAN infrastructure components is necessary. He can also profit from the flexibility and matureness of the Linux kernel [125] network stack. An example of such software AP product is the Linksys WRT54g access point [89], which is fully based on the Linux kernel (2.4) with complete firmware source code and a development kit available[5]. A software access point based on the Linux kernel version 2.4 and a Prism [69] chip-set wireless card was also my preferred configuration for conducting WLAN experiments and measurements.

3.3 Description of IEEE 802.11 Layers

As any 802.x like protocol the initial 802.11 standard covers the MAC as well as the radio PHY layer. The standard currently defines a single MAC type which interacts with three different PHYs (all of them running at least at 1 and 2 Mbit rates): Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) both in the 2.4 GHz band and also an InfraRed PHY on a near-IR channel.

3.3.1 802.11 PHY Layer

Lower Limit	Upper limit	Regulatory range	Geography
2.402 GHz	2.480 GHz	2.400–2.4835 GHz	North America
2.402 GHz	2.480 GHz	2.400–2.4835 GHz	Europe[a]
2.473 GHz	2.495 GHz	2.471–2.497 GHz	Japan
2.447 GHz	2.473 GHz	2.445–2.475 GHz	Spain
2.448 GHz	2.482 GHz	2.4465–2.4835 GHz	France

Figure 3.4: Frequency bands available for 802.11(b) WLANs

The PHY layer plays a central role in the actual delivery of wireless network packets, since a wireless transaction is nothing else than a transmission and reception of electromagnetic waves (same holds for infrared which is also an electromagnetic wave but with a much higher frequency than 2.4GHz microwaves). The original IEEE standard defined two radio based transmission techniques as well as one in the infrared frequency band.

The DSSS type of PHY modulation provides a wireless LAN with both 1 Mbit/s

[4]The majority of low-cost 802.11 AP devices are consumer wireless gateways and routers offering DSL line support.

[5]The openess of the system manifestated in a variety of available firmware images e.g. DD-WRT, OpenWRT, Alchemy and EWRT to just name few.

and 2 Mbit/s data payload communication capability and works by chipping the baseband microwave signal at 11 MHz with a 11-chip PN code. The DSSS system uses baseband modulations of differential binary phase shift keying (DBPSK) and differential quadrature phase shift keying (DQPSK) in order to provide the 1 Mbit/s and 2 Mbit/s data rates respectively.

The FHSS PHY operates at a rate of 1Mbps using two-level Gaussian frequency shift key (GFSK) modulation while at 2 Mbit/s the modulation scheme is changed to four-level Gaussian frequency shift keying (4GFSK). The number of transmit and receive frequency channels used for hopping is 79 for the USA and Europe and 23 for Japan. The available operation frequencies by geographical region are shown in the figure 3.4.

The IR PHY uses near-visible light in the 850 nm to 950 nm range for data transmission. Unlike many other infrared devices (e.g. serial IR available in portable PCs) the 802.11 IR PHY is not directed. That means that the receiver and transmitter devices do not have to be aimed at each other and do not need a clear line-of-sight. This central feature of 802.11 permits the construction of a true LAN system, while with an aimed system it would be difficult or impossible to install a LAN because of physical constraints. A pair of conformed infrared devices would be able to communicate in a typical office environment at a range of up to 10 m, however under favourable conditions it may increase to roughly 20 m.

The two radio spread spectrum techniques defined by the 802.11 standard provide a high level of robustness against channel noise and interferences and also permit neighbouring wireless cells to share parts of the radio spectrum without explicit cooperation[6].

The DSSS type of modulation exhibits higher level of robustness compared to the FHSS modulation, even when FHSS uses twice the transmitter power output level. Regardless of the large number of hop frequencies defined for FHSS in the 802.11 standard, adjacent channel interference behaviour limits the number of independently operating collocated systems. Hop time and smaller maximum packet size introduce more overhead into FHSS compared to DSSS and therefore affect the maximum achievable throughput.

Although FHSS is less robust than DSSS, it provides a more graceful degradation in throughput and connectivity under interference or weak signal conditions, since it will continue to work over some of the hop channels a little longer than over the other hop channels. DSSS however, may still deliver a reliable link over a distance where only few FHSS channels would be working, but at the other hand it may fail suddenly, if the signal to noise ratio (SNR) drops below some threshold[7]. For a wireless network infrastructure with multiple access points covering a large area, DSSS gives a higher potential throughput with fewer access point compared to FHSS therefore lowering the overall cost of the installation. A good comparison of the original 802.11 PHYs can be found in the article [76].

[6]**Since 802.11b channel frquency bands overlap this is even required.**

[7]**The limit for most of the radio units so far seen by the author seems to be -90dBm (total of antenna and radio gains) for the lowest possible rate of 1Mbps.**

3.3.2 802.11 MAC Layer

Beyond the standard functionality usually performed by MAC layers, the 802.11 MAC performs other functions which are typically related to upper layer protocols (e.g. IP or TCP) such as fragmentation and defragmentation, packet retransmissions and positive acknowledgements. The 802.11 MAC layer defines two different access methods, the Distributed Coordination Function (DCF) and the Point Coordination Function (PCF) described below.

Basic Access Method: CSMA/CA

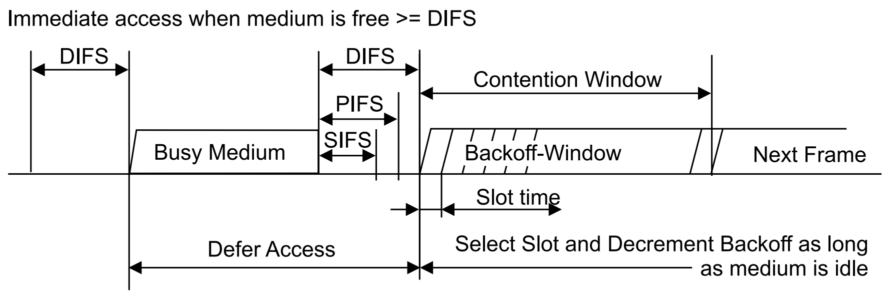
Immediate access when medium is free >= DIFS
DIFS
DIFS
Contention Window
PIFS
SIFS
Busy Medium
Backoff-Window
Next Frame
Slot time
Defer Access
Select Slot and Decrement Backoff as long as medium is idle

Figure 3.5: Basic CSMA/CA channel access procedure.

The primary channel access procedure of 802.11 called Distributed Coordination Function (DCF) is a Carrier Sense Multiple Access method with Collision Avoidance mechanism (usually known as CSMA/CA). CSMA protocols are well known in the industry, where the most popular is probably the Ethernet [39], which is a CSMA/CD type protocol (CD standing for collision detection). The 802.11 CSMA protocol works as follows. A station desiring to transmit a packet must sense the radio channel for some minimum time period T. If the medium is busy (e.g. some other station is already transmitting), then the station will defer transmission to a later time (that is, back off). On the other hand, if the medium is sensed free, then the station is permitted to transmit immediately as shown in the picture 3.5.

These kind of protocols is sufficiently effective if the medium is not heavily loaded. It allows stations to transmit some amount of packets with minimum delay, however there is always a risk of stations transmitting at the same time (collision), if more than one station sense idle channel and decide to transmit at once. In this type of protocol one must also consider the time needed for the signal from transmitting station to reach all participants in the network, because the speed of an electromagnetic wave is limited. That means that a station sensing the wire(less) may not yet hear another already transmitting station because the electromagnetic wave did not reach the sensing station yet. Therefore a sensing station may erroneously assume, that the medium is free and also start a transmission. The duration of the minimum sensing time T limits the maximum coverage area of a WLAN by the relation $d = c\dot{T}$, where c the speed of light. For 802.11b it is roughly 6km, which is far above the limit imposed by the transmitter output power. Note that this limit also apply to directed WLAN links (otherwise collisions will significantly degrade the link quality for high loads).

These potential collision situations must be identified and retransmission performed by the CSMA MAC and not by upper layers, since upper-layer retransmission would introduce an unacceptably large delay. In case of wired Ethernet for example, a collision is recognised by the transmitting stations, because they can simultaneously send and listen to their own transmission (an Ethernet station can loop back its own signal from the wire into the receive circuitry and compare it with the output signal). If a collision situation is detected Ethernet stations go to a retransmission phase based on an exponential random backoff algorithm. While these collision detection mechanisms are a good idea on a wired LAN, they cannot be used on a Wireless LAN environment for three main reasons:

- implementing a collision detection mechanism would require the implementation of full duplex radio units capable of transmitting and receiving at the same time, a feature which would significantly raise the price of consumer wireless cards,
- in a wireless environment we cannot assume that all stations can hear each other (which is the basic assumption of the collision detection scheme described above). Therefore a station going to send a packet and sensing an idle transmission medium, can not assume that the channel is free around the receiver area too,
- in opposite to wireline transmission, the strength[8] of the radio signal from a wireless station shows a strict R^{-2} dependence (in case of an omnidirectional antenna) well known from the physics of electromagnetic waves. That means that the signal strength is highest at the sender location[9] but drops rapidly with distance. Even if equipped with a full duplex radio unit, a transmitting node may not be able to hear another wireless transmission, since the radio signal at its position is dominated by its own transmission, while being the other way around at the location of the second station[10].

In order to mitigate above problems, the 802.11 MAC uses a Collision Avoidance (CA) mechanism together with a positive acknowledge scheme which works as follows. A station wishing to transmit a packet senses the medium and if the medium is busy, then it defers and follows the backoff procedure as described above. If the medium is free for a certain time period (called DIFS for Distributed Inter Frame Space), then the station can directly transmit the packet. The receiving station will compute the CRC (cyclic redundancy code) checksum of the received data frame and compare it with the value of the CRC field in the received packet and if they are equal, send a short acknowledgement packet (ACK) back to the sender. This is an autonomous MAC function performed in hardware, because the timeout for the acknowledgement frame to arrive is very short (SIFS for Short Inter Frame Space).

[8]or more precisely the energy density per reference volume

[9]it is slightly more complicated since there is so called near and far field area around electromagnetic sender.

[10]in case of wired Ethernet the signal strength does not depend so much on the location of the receiver because damping of electromagnetic waves in a cable is only moderate.

Reception of the acknowledgement will indicate to the transmitting node, that no collision occurred and that the packet arrived at the destination station without corruption.

If the sender does not receive the ACK frame before the acknowledgement timeout expires, the data packet is assumed to be lost (it is possible that the acknowledgement gets lost or corrupted, however for the MAC it doesn't make a difference) and the wireless node will have to retransmit the actual packet until it gets acknowledged or is dropped after a maximum number of retransmissions. The 802.11 standard defines short and long (with respect to the RTSThreshold MAC variable defining the byte-length threshold at which the long retry limit is used) packet retry counts for this purpose.

The IEEE 802.11 standard defines also a Virtual Carrier Sense mechanism in order

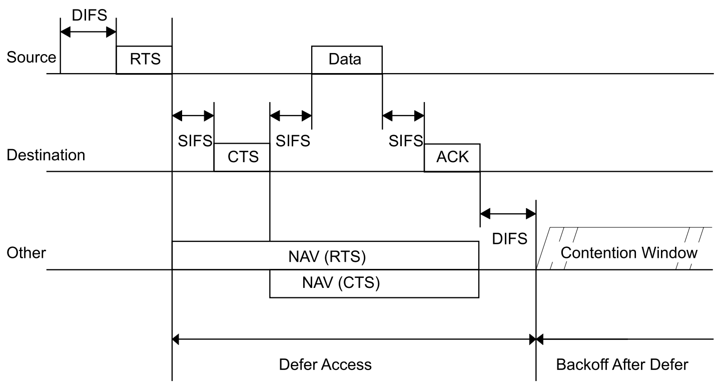

Figure 3.6: IEEE 802.11 virtual carrier sense using RTS/CTS (request- and clear-to-send).

to reduce the probability of collisions between two or more stations if they cannot hear each other (so called `hidden terminal` problem) and it works as follows. A station willing to transmit a data packet will first transmit a short control packet called RTS (Request To Send), which includes the source and destination address as well as the total duration of following wireless transaction (that is, the next packet and its respective ACK).

Upon successful reception of the RTS, the destination station will transmit a short response control packet called CTS (Clear To Send), which will include similar duration information. All stations receiving either the RTS and/or the CTS will set their virtual carrier sense indicator (called `NAV`, for `Network Allocation Vector`) to the given duration and will use this information together with the physical carrier sense, while waiting for an idle wireless medium as shown in the figure 3.6.

This mechanism reduces the probability of a collision in the receiver area by a station, which is "hidden" from the transmitter (which can be heard at the destination but not at the source of wireless transmission) only to the short duration of the RTS packet, because the station will hear the CTS sent from the destination node and reserve the medium as busy until the end of the wireless transaction. The duration information in the RTS packet also protects the transmitter area from collisions during the transmission of the MAC level ACK (by stations which are out of range from the acknowledging station).

It should also be noted, that since the RTS and CTS are both very short frames[11], the virtual carrier sense mechanism reduces the overhead of collisions, because they are recognised much faster than if the complete packet had to be transmitted and the collision situation recognised by absence of the final ACK frame. However this is only true if the wireless packet is significantly bigger than the RTS frame. Therefore the 802.11 standard allows for sufficiently short packets to be transmitted without the RTS/CTS virtual carrier sense and can be controlled by setting a MAC parameter called RTSThreshold.

While the RTS/CTS mechanism delivers sufficient protection from the hidden terminal problem, there are other issues in a wireless type network related to network topology and commonly referred as `exposed terminal`, `moving terminal`, `temporarily deaf terminal`, `heterogeneous terminal` and `sleeping terminal` problem [138].

The 802.11 Backoff Algorithm

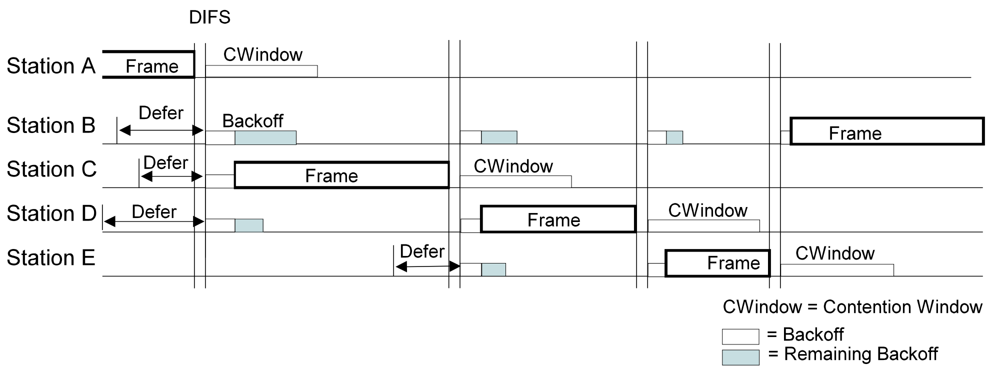

Figure 3.7: Subsequent 802.11 frame transfers with backoff.

Backoff is a well known method to resolve contention between different stations trying to access a shared transmission channel at once. The 802.11 backoff procedure requires each station to choose a random number N between 0 and some maximum value M, and then wait for this number of time slots before actually trying to occupy the wireless channel. The interval $[0, M]$ used for selecting random numbers is called `contention window` (CW). During the backoff phase a wireless station is supposed to regularly check (e.g. at the beginning of each time slot) the status of the transmission channel. The duration of the slot time (ST) is defined in such a way, that a station will always be capable of determining if another station has accessed the medium in the previous time slot. The 802.11 draft states that a waiting station must freeze its backoff timer if it senses busy medium before its backoff timer expires. That means that in the next channel access cycle, all stations already executing the

[11] **with respect to the duration of an average data frame.**

backoff procedure will have their timers (that is the number of time slots to wait) reduced by the number of slots they already have waited in the previous cycle. This type of backoff mechanism is called `residual backoff`.

In case of IEEE 802.11 wireless network an exponential backoff procedure is also applied if necessary. Exponential backoff means, that each time the station chooses a random backoff slot and collides at the end of the backoff period, it will increase its actual maximum value M used for slot selection exponentially, doubling the CW each time until some absolute maximum CW value is reached. Current 802.11b products use a value of 32 slots for the initial minimum CW and a value of 1024 slots for the absolute maximum CW.

The 802.11 standard requires that the above compound backoff algorithm must be performed in the following cases:

- when the station senses busy medium after a MAC idle period (that is if the station did not have a packet to send for a while)
- after each successful transmission of a packet while station's MAC is still back-logged
- and after each wireless retransmission

The only case when the backoff mechanism is not used by wireless station is when a new packet is passed to the MAC after an idle period and the medium is sensed free for the duration of the DIFS. This permits a station which has rarely a packet to send (e.g. an interactive SSH session) to transmit with minimum delay. The figure 3.7 shows a sequence of wireless transactions with multiple stations deferring transmissions due to busy medium.

Inter Frame Spaces

The 802.11 standard defines four types of Inter Frame Spaces, which are used to provide different transaction priorities over the wireless:

- SIFS - which stands for Short Inter Frame Space. It is used to separate single chunks belonging to same wireless dialogue (e.g. fragment-ACK) and is also the minimum Inter Frame Space (silence) which can appear on the wireless channel. As long as there is an ongoing transmission and its fragments are separated by SIFS, other stations cannot seize the wireless channel (if they can hear each other), since they are required by the standard to sense idle channel for a period longer than SIFS. This value is fixed per PHY and has been calculated in a manner, that a transmitting station will be able to switch back to the receive mode and be capable of decoding of an incoming fragment (e.g. ACK). For example on the 802.11 FH PHY this value is set to 28 microseconds
- PIFS - Point Coordination IFS, which is used by the access point node (called also Point Coordinator in this case) to gain priority access before any other regular (that is non-AP) wireless station. This value is longer than SIFS but shorter than DIFS, giving the point coordinator priority over regular stations in capturing an idle channel

- DIFS - Distributed IFS, is the Inter Frame Space used by a station to start a new wireless transmission and is longer than the two previous IFS intervals
- EIFS - Extended IFS, which is a special longer IFS used by a station which has received a packet it could not fully understand. This is required to prevent a station (which could not understand the duration information for the Virtual Carrier Sense) from colliding with future packet(s) belonging to the current wireless dialogue

MAC Level Acknowledgements

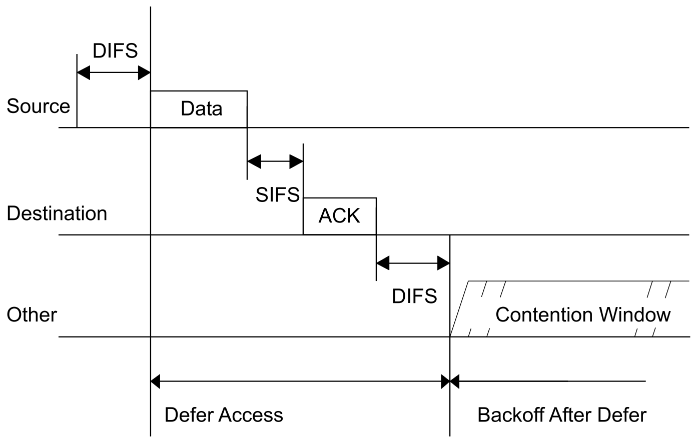

Figure 3.8: Timing of 802.11 conversation with MAC level acknowledgement.

As already mentioned, the MAC layer performs collision detection by expecting the reception of a positive acknowledgement for every fragment it transmits (an exception to these are packets which have more than one destination, such as multicast or broadcast packets, which are not acknowledged). The lack of reception of an expected ACK frame indicates to the source station, that a transmission error has occurred. After a successful reception of a frame requiring an acknowledgement, transmission of the ACK frame must begin after the SIFS period without regard to the busy/idle state of the medium. Note however, that the destination node may have received the frame correctly, and that the error may have occurred in the reception of the ACK frame. To the initiator of the frame exchange, this condition is indistinguishable from an error occurring in the initial frame.

Packet Fragmentation and Reassembly

Typical LAN protocols use packets of several hundreds of bytes (e.g. standard 100Mbps Ethernet longest packet could be up to 1518 bytes long where up to 1500 bytes are available to upper level protocols like IP), on a Wireless LAN environment there are some reasons why it would be preferable to use smaller packets:

- because of the relatively high bit error rate (BERR) of a radio link, the probability q for a packet to get corrupted increases with the packet size. For a given single bit error probability p and packet length N in bits, the corruption probability

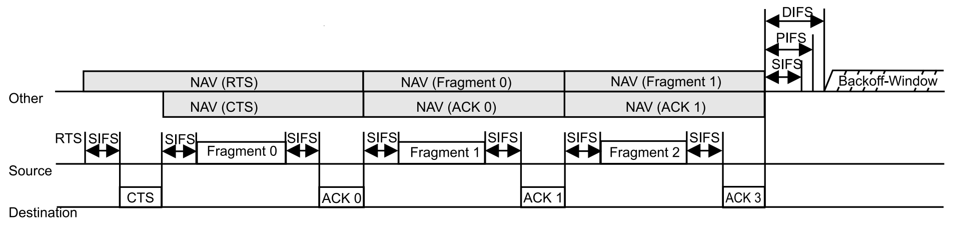

Figure 3.9: Complete 802.11 transaction with virtual carrier sense and frame fragmentation.

evaluates to $q = (1-p)^N$ that is in first order approximation $q = N(1-p)$. In case of packet corruption (regardless of if it was a collision or just noise), the smallest the packet the less overhead it causes if retransmitted.

- on a frequency hopping system the medium is also interrupted periodically for hopping (in case of 802.11 every 20 milliseconds), so the smaller the packet, the smaller the chance that the transmission will be postponed.
- the smaller the packet the shorter the time to resolve wireless collisions, since a 802.11 wireless collision can only be detected at the end of the frame exchange.

On the other hand it doesn't make sense to introduce a new LAN protocol which cannot deal with maximum packet size of 1518 bytes used on wired Ethernet, so that the committee decided to go around the problem by adding a simple fragmentation/reassembly mechanism at the MAC layer. The fragmentation mechanism is a simple send-and-wait algorithm, where the transmitting station is not allowed to transmit a new fragment until:

- the station receives an ACK for the actual outstanding fragment
- the sending station decides that the fragment was retransmitted too many times and then drops the whole frame where the fragment belongs to

It should also be noted that the standard does allow a station to transmit to a different MAC address between retransmissions of a given fragment. This may be very useful when a station (probably an AP) has several outstanding packets for different destinations and one of them does not respond for a while. Therefore a dead station can not block the access point node and starve the packet flow in the wireless network. The figure 3.9 shows a fragmented wireless transaction.

Synchronisation Inside a Wireless Cell

Wireless stations usually need to keep synchronisation. This is needed for keeping hopping synchronised but also other functions like power saving and the backoff procedure. In an infrastructure BSS this task is performed by all the stations updating their clocks according to the AP's clock using the following mechanism. The AP transmits periodic frames called `Beacon Frames`. These frames contain the value of the AP's clock at the moment of the transmission (note that this is the moment when the transmission really occurs, and not when it is put in the queue for transmission, since the `Beacon Frame` is transmitted using the rules of CSMA and the transmission may be delayed significantly). The receiving stations check the value of their clock at the receiving moment and eventually correct it to stay synchronised with the AP's clock. This prevents clock drifting which could cause loss of synch after a couple of hours.

It is also worth to note that the periodic Beacon frames include management information like supported rates, header length, encryption status etc. and may also include vendor-specific extension tags.

Power Saving Mode

Wireless LANs are typically related to mobile applications and for such applications battery power may be a scare resource. For this reason the 802.11 network standard directly addresses the issue of power saving and defines a complete mechanism which allows stations wishing so to go into sleep mode even for long time periods without missing network frames directed to them.

The main idea behind the power saving mechanism is that the AP which a station has associated with, maintains an up-to-date record of the stations currently working in power saving mode and buffers all the packets addressed to these stations until either the stations specifically require to get its packets by sending a polling request or until they change their operation mode to active.

The AP also transmits periodically (as a part of its beacon frames) information about which of the power saving stations have frames buffered at the AP. The power saving stations should wake up periodically in order to receive one of these beacon frames and if there is an indication that there is a frame stored at the AP then the station should stay awake and send a poll message to the AP to get its frames delivered. Also multicast and broadcast messages are stored by the AP and transmitted at a pre-known time (on 802.11 those frames are delivered each 100 milliseconds which is the default beacon interval for 802.11b devices), where all power saving stations who wish to receive this kind of frames should be awake.

Cell Association, Authentication and Roaming

When a station wants to access an existing BSS (either after power-up, sleep mode, or just if entering the BSS area) the station needs to get synchronisation information from the Access Point (or from the other stations when the network is operating in ad-hoc mode, which will be discussed later). The wireless node can obtain this information by one of the two means:

- passive AP scanning: in this case the station just waits to receive one of beacon frame from the AP responsible for the wireless cell

- active scanning: here the station tries to find an access point by transmitting probe request frames and waiting for a probe response frame from an AP in range. The two methods are both valid on an IEEE 802.11 network and either of the two can be chosen according to the power consumption/performance trade-off

Once a station has found an available access point and decided to join its BSS, it will go through an authentication process, which is the interchange of information between the AP and the station, where each side proves the knowledge of a secret password. An open-system mode where any station is permitted to join the BSS is defined in 802.11. In the open-system mode there is still a possibility to use a MAC level filter to allow/deny certain stations and the two things should not be confused.

After a successful authentication, a wireless client may start the association process, which is the exchange of information about the station's and BSS capabilities and which allows the DSS (the set of APs) to know about the current position of the station and eventually modify network routes. Only after the association process is completed a WLAN station is capable of transmitting and receiving data frames over wireless.

Since wireless is a very dynamic medium, the 802.11 standard defines also a mechanism called roaming. Roaming is the process of moving from one cell (or BSS) to another without loss of wireless connectivity. This function is very similar to the cellular phones handover with two main differences:

- in a LAN like system which is packet based, the transition from cell to cell may be performed between packet transmissions, as opposed to telephony, where the transition may occur in the middle of a phone conversation and this makes the LAN roaming an easier task
- for a voice system a temporary disconnection may not affect the conversation quality since humans do not perceive short interruptions if they are not longer than some physiological threshold[12]. In contrast for a packet based environment interrupted connectivity may significantly reduce application performance, if the loss of connectivity is passed up to any upper layer protocols (e.g. TCP).

The 802.11 standard does not exactly describe, how roaming should be performed and only defines the basic building blocks of the process itself. This includes the active/passive scanning and a re-association process, where a station roaming from one access point will become associated with a new one.

Point Coordination Function (PCF)

Beyond the basic mode of operation based on the distributed coordination function there is an optional Point Coordination Function mode defined in the IEEE 802.11 standard, which may be used to implement time-bounded services like voice or video transmission and provide better quality of service (QoS) to the participants in the wireless network.

The point coordination function makes use of the higher priority available to the access point by switching to the smaller inter-frame spaces reserved for the PCF mode (PIFS) while trying to capture the channel. By using this higher priority the

[12]Human senses like ears are very imperfect and this fact is used broadly in communication systems like telephony where the frequency spectrum of transmitted speech may be significantly reduced.

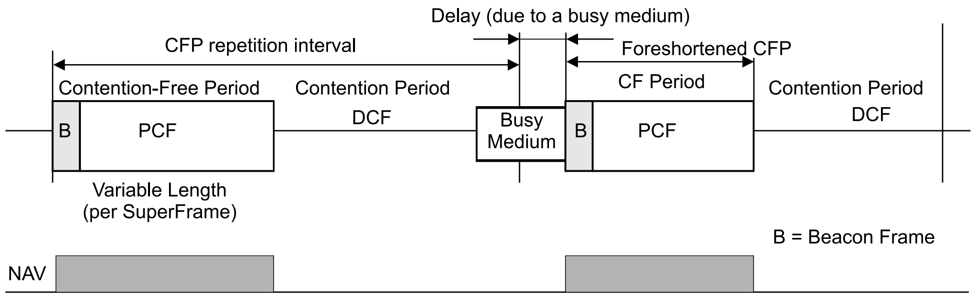

Figure 3.10: Coexistence of DCF and PCF in the 802.11 superframe.

access point may issue polling requests to associated stations to trigger their data transmission and deliver its own frames to these stations, hence centrally controlling the access to the wireless medium. However, in order to provide also a transmission chance to legacy stations, which are unaware of the PCF mode of operation (or to stations which did not associate with the AP yet), there is a provision, that the access point must leave enough time for at least one DCF-like transaction between two consecutive PCF contention-free periods. The combined time interval where the DCF and the PCF mode of operation coexist is called `super-frame` in the 802.11 nomenclature and (potentially) starts with each beacon frame transmitted by the cell coordinator (AP).

The point coordinator (PC) in the cell may terminate the contention free period (CFP) at any time based on the amount of traffic and the size of its polling list. Since the transmission of any beacon frame may be delayed due to a busy-medium condition at the regular (that is periodic in majority of 802.11 products and usually set to 100ms) beacon transmission time, the CFP interval may also be foreshortened by the amount of the delay. In case of a busy channel due to DCF traffic, the beacon frame is delayed for the time required to complete the ongoing DCF frame exchange and the AP will use PIFS to immediately seize the channel at the end of the current transaction. A sample super-frame where simultaneous PCF- and DCF-type access takes place has been shown in the figure 3.10.

Despite its theoretical superiority for a number of applications like voice over IP, the PCF mode has not been implemented in the great majority of up-to-date 802.11-conformant wireless products. The standard also does not describe exactly how PCF polling should be performed. The reason is just simple: it is too difficult to define an algorithm suitable for all types of applications. Moreover even if the requirements of the wireless stations imposed on traffic scheduling are reduced to just very few parameters (like: there is some amount of real-time traffic from each node which needs a bandwidth-delay guarantee and the remaining cell capacity should be served in a best-effort manner), it may still be a rather complex optimisation problem difficult to implement in hardware[13].

[13]**for the same reason I'm convinced that future WLAN products should include a programmable hardware layer where users may load extensions of the basic card firmware. They should include a real-time programming language which permits implementation of various MAC protocol extensions of low to middle complexity.**

There is an essential amount of research already done in the area of PCF-like network access, for example in the article [82] an evaluation of a PCF-like wireless access scheme in presence of real-time constraints is presented. The authors report that the channel bandwidth utilisation compared to pure DCF mode of operation has been optimised, while also reducing the mean packet wait time for real-time flows. However, the main problem of a PCF access scheme is the overhead imposed by the signalling protocol necessary to provide sufficient polling information to the point coordinator. The article reports that about 11 percent of the channel capacity is wasted for polling, if 16 concurrent stations with one real-time flow per station, 64kbit/s each, must be served in PCF mode. The signalling protocol used is an explicit notification scheme via dedicated signalling frames sent during the DCF period of the super-frame. However, an improved scheme using implicit signalling at the MAC layer is proposed in the article and reduces the polling overhead to about 4 percent of the channel bandwidth.

Another related PCF-like access scheme is Blackburst described in the article [120] and whose main goal is the minimisation of delay for real-time traffic. Blackburst requires that all high-priority stations must try to access the channel at constant intervals of the length T and must also have the ability to jam (that is to prevent the other stations in range from transmitting) the medium for a defined period of time, while low priority stations may use the ordinary 802.11 DCF method for channel access. If a station wishing to send real-time data senses busy medium, it waits like in the DCF operation mode until the channel becomes idle, but then immediately enters a Blackburst contention period jamming the channel for the defined period of time, which is determined by the amount of time the station has been waiting while the medium was busy. After the Blackburst jamming period the station listens to the channel again in order to check if another station is still sending a longer Blackburst signal implying that it has been waiting longer. Finally the station sending the longest Blackburst signal wins the contention period and is allowed to send its packet. By using slotted (quantised) time and imposing a minimum frame size on real-time frames it can be guaranteed that each Blackburst period yields an unique winner. In Blackburst after each successful frame transmission next real-time frame is rescheduled after the time period T and therefore real-time flows from different nodes synchronise and can then cross the wireless without collisions.

802.11 Ad-hoc Networks

For a number of applications users will desire to build up wireless networks without any infrastructure (or even without an Access Point device). This may include file transfer between two notebook users, a coworkers' meeting outside the office, or even military applications for the battlefield, etc. The 802.11 standard addresses this need by the definition of an "ad-hoc" mode of operation. In this case there is no access point and part of its functionality is performed by the end-user stations (like beacon generation, synchronisation, etc.), while other functions are not supported (like frame-relaying between two stations not in range or power saving mode).

An important difference between the managed mode and the ad-hoc mode of operation is that in the first case wireless nodes are obliged to always talk to the access point node, even if the final destination of the packet is a direct neighbour in the same cell. Therefore there is a doubled overhead, if two associated stations must talk to each other. In opposite, the ad-hoc mode permits every wireless station to directly send packets to each destination node in radio range providing a real any-to-any and one-to-many network. The CSMA/CA access method used at the MAC level is independent of the actual mode of operation.

3.3.3 Overview of 802.11 Frame Types

In the previous sections I have described the basic structure of a 802.11 wireless LAN, its logical layers and modes of operation. Since I want to get closer at the problem of quality of service in a wireless LAN later on in this work and quality of service can not be decoupled from details of frame transmission in the network, I want now to provide an abbreviated description of 802.11 MAC level frame types and their meaning.

Basically there are three main types of frames in a 802.11 like wireless network depending on their logical function:

- data frames: are used for data transmission and carry payload from higher level protocols like IP
- control frames: are used to control the access to the medium (e.g. RTS, CTS, and ACK frames)
- management frames: which are frames similar to data frames, used to exchange management information but not forwarded to upper layers. Management frames are subdivided into different subtypes, according to their specific function.

General Frame Structure

Despite the type all of 802.11 frames consists of following four building blocks: `preamble`, `PLCP Header`, `MAC data` and `CRC checksum` as shown in the picture 3.11 and described below.

The frame preamble is PHY dependent and includes a synchronisation filed which

PLCP Preamble		PLCP Header			Whitened PSDU
Sync	Start Frame Delimiter	PLW	PSF	Header Error Check	
80 bits	16 bits	12 bits	4 bits	16 bits	Variable number of octets

Figure 3.11: Structure of the physically transmitted 802.11 frame. The variable length MAC Protocol Data Unit (MPDU) contains the LL (Link Layer) frame.

is a 80-bit sequence of alternating zeros and ones, which is used by the PHY circuitry to select the appropriate antenna (if antenna diversity is used[14]) and to adjust

[14]many products offer multiple antennas selecting that one providing the best signal quality automatically. Recent 802.11 innovations also include MIMO technology with multiple antennas which can help to combat interference problems e.g. in buildings.

the steady-state frequency offset to the sender of the packet. Since the clocks and electronic oscillators of even equal 802.11 devices tend to differ, a PLL circuit is provided in hardware to synchronise the receiver's clock. The preamble also includes a so called `SFD` (Start Frame Delimiter) which consists of the 16-bit binary pattern 0000 1100 1011 1101, which is used to define frame timing.

The 802.11 PLCP header is always transmitted at a rate of 1 Mbit/s, since the remaining part of the packet may be transmitted at different speeds. A constant bit rate PLCP permits the receiver to read the rate and to reprogram its PHY unit for the remaining part of the transmission. The PLCP header contains a PLCP_PDU length word, which defines the number of bytes contained in the packet and is useful for the PHY to correctly detect the physical end of the packet. Moreover it contains also a PLCP signalling field, which currently only specifies the rate information for the packet body encoded in 0.5 Mbps increments. The last field of the PLCP header is the header error check field, which is a 16 bit CRC protecting header data from transmission errors.

The remaining part of the PHY frame carries the MAC data and has the general

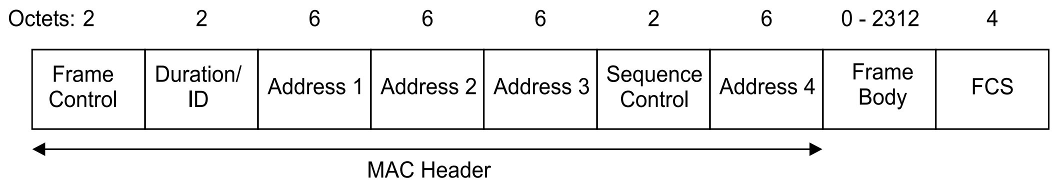

Figure 3.12: Format of the 802.11 LL frame.

format shown in the figure 3.12 consisting of a frame control, duration, address 1-4, sequence control, variable length frame body and a data CRC checksum field (or frame checksum, FCS). The bits in the frame control word decide which of the MAC frame fields are valid and define the exact type of the wireless frame.

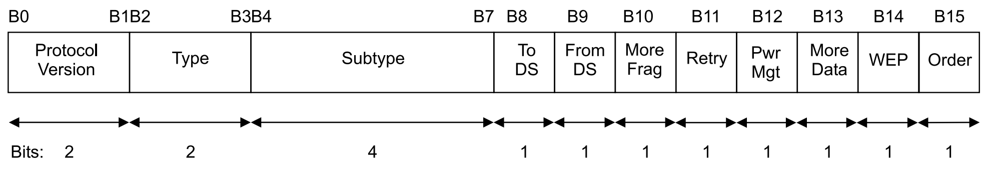

Figure 3.13: Structure of the frame control field of the 802.11 LL frame.

The `protocol version` field consists of 2 bits, which are guaranteed to be invariant in size and placement across any following version of the IEEE 802.11 standard and can be used by 802.11 devices to recognise possible future versions. In the current version of the standard this field is set to 0.

The two bit `type` field decides whether the wireless frame is a management (bit combination 00), a control (bit combination 01) or a normal data frame (10), while the bit combination 11 is reserved. The `subtype` field select the exact function of the frame.

A 802.11 MAC frame may contain up to 4 addresses[15] depending on the setting of the `ToDS` and `FromDS` bits defined in the `control` field as described below. The format of the `address` fields is compatible with the Ethernet standard, that is they are composed of six octets.

`Address-1` is always the next recipient address (i.e. the station on the BSS who is the immediate recipient of the packet). If `ToDS` bit is set, this must be the address of the AP station and if `ToDS` is not set, then this is the address of the end station.

`Address-2` is always the transmitter address (i.e. the station who is physically transmitting the packet). Therefore if `FromDS` is set, it is the address of the AP and if it is not set then it is the address of the source station.

`Address-3` is in most cases the remaining (missing) address, so that on a frame with `FromDS` set to 1, it is the original source address and if the frame has the `ToDS` bit set, then `Address-3` is the destination's node address.

`Address-4` is used in the special case, where a wireless distribution system (WDS) is used and the frame is being transmitted (relayed) from one access point to another. In this case both the `ToDS` and `FromDS` bits are set, so that both the original destination and the original source addresses must be transmitted.

The `duration/ID` field is 16 bits in length and has two meanings depending on the frame type: in power-save polling messages this is the station ID and in all other frames this is the duration value used for virtual channel allocation via NAV. The content of this field is as follows: in control frames of subtype `power save` the `duration/ID` field carries the association identity (AID) of the station which transmitted the frame in the 14 least significant bits (LSB), with the 2 most significant bits (MSB) both set to 1. The value of the AID is in the range 1 to 2007. In all other frames, the `duration/ID` field contains transaction duration value. For frames transmitted during the contention-free period, the duration field is set to the value of 32 768. Whenever the contents of the `duration/ID` field are less than 32 768, the duration value is used to update the network allocation vector according to the virtual carrier sense mechanism already described.

The `sequence control` field is used to represent the order of different fragments belonging to the same MAC frame and also to recognise packet duplications. It consists of two subfields: `fragment number` and `sequence number`, which define the frame ID and the logical number of the fragment inside a bigger frame.

The `more fragments` bit is set to 1 in the frame header when there will be more fragments belonging to the same frame following this current fragment, whereas the `more data` field is related to the power save mode. It is set to inform a station currently in power-save mode that more MSDUs are buffered for that station at the AP. The `more data` field is valid in directed (that is unicast) data or management type frames transmitted by an AP to a station in power-save mode.

A `more data` value of 1 indicates, that at least one additional buffered MSDU or MMPDU is present for the same station. The `more data` field may be also set to

[15] in contrast to Ethernet where only destination and source addresses are needed.

1 in directed data type frames transmitted by a station during the contention-free period to the point coordinator in response to a CF-poll frame, to indicate, that the station has at least one additional buffered MSDU available for transmission. The `more data` field is set to 0 in all other directed frames.

The `more data` field is set to 1 in broadcast/multicast frames transmitted by the AP, when additional broadcast/ multicast MSDUs, or MMPDUs are waiting for transmission at the AP during the actual beacon interval. The `more data` field is set to 0 in broadcast/multicast frames transmitted by the AP, when no more broadcast/ multicast MSDUs, or MMPDUs remain to be transmitted by the AP during this beacon interval and in all broadcast/multicast frames transmitted by non-AP stations.

The `power management` field is 1 bit in length and is used to indicate the power management mode of a transmitting station. The value of this field remains constant in each frame from a particular station within a single frame exchange. The value indicates the mode in which the station will be after successful completion of the actual frame exchange sequence. A value of 1 indicates that the sender will be in power-save mode. This field is always set to 0 in frames transmitted by an AP. A complementary function plays the `more data` bit and it is set by the AP to indicate, that there are more frames buffered for this station. The station may decide to use this information to continue polling of buffered data or even change the operation mode to active.

The `retry` bit indicates that this fragment is a retransmission of a previously transmitted fragment and this may be used by the receiver station to recognise duplicate transmissions of frames which may occur when an acknowledgement frame is lost.

Finally the `CRC` is a 32 bit field containing 32-bit Cyclic Redundancy Checksum computed over the frame body and the `WEP` bit indicates, that the body is encrypted according to the WEP algorithm (not discussed here).

3.4 Related Wireless Standards

3.4.1 HiperLan and HiperLan2

The history of the HiperLan standardisation is the total opposite of 802.11. HiperLan was designed by a committee of researchers organised within the ETSI (European Telecommunications Standards Institute) without strong vendors influence and is quite different from existing wireless products. This historical background yielded in 1996 a more uniform standard than 802.11 but with some advanced and superior features.

One of the advantages of HiperLan is that it works in a dedicated frequency band (5.1 to 5.3 GHz allocated only in Europe) and therefore doesn't need to include advanced and more costly spread spectrum techniques. The PHY signalling rate is 23.5 Mbit/s on one of 5 fixed channels. The protocol uses a variant of CSMA/CA called EYNPMA (Elimination Yield - Non-preemptive Priority Multiple Access [132]) based on packet time to live and priority and includes (like 802.11) MAC level retransmissions providing a reliable MAC layer. The protocol includes optional encryption (but with no cipher algorithm mandated so far) and a power saving mode similar to 802.11.

A superior feature of HiperLan over IEEE's 802.11 is its integrated ad-hoc routing algorithm: if a destination is out of direct range, intermediate nodes will automatically forward its packets through the optimal route within the HiperLan network (the routes are periodically recalculated). Hiperlan is also totally ad-hoc, requiring no configuration and no central controller device.

The main drawback of the HiperLan standard is that it doesn't provide real isochronous services (but comes quite close with its time to live and packet priority), doesn't fully specify an access point and portal mechanisms and also hasn't really been proven to work well on large scale in the real world. Transmission overhead tends also to be quite large due to relatively big packet headers.

While the first version of the HiperLan standard was designed to build ad-hoc networks, the second edition of HiperLan [112] addresses managed infrastructure and wireless distribution systems. HiperLan2 has been the first standard to be based on the OFDM modulation type. Each sub-carrier may use different modulation type (and different convolutional code, a sort of FEC, that is Forward Error Correction), which allow to offer multiple bit-rates (6, 9, 12, 18, 27 and 36 Mb/s, with optional 54 Mb/s) with expected real-world performance of around 25 Mb/s. The OTA physical channel width is 20 MHz and includes 48 OFDM carriers used for data and 4 additional carriers, used as references (pilot carriers, total is 52 carriers with 312.5 kHz spacing).

From the logical point of view HiperLan2 is a connection oriented Wireless ATM system and the MAC protocol is a centrally coordinated TDMA scheme with reservation slots. Each slot has a 54 byte payload and the MAC provides SAR (Segmentation and Reassembly, that is autonomously fragments large packets into 54 byte cells) and ARQ (Automatic Request, MAC level retransmissions). The scheduler (in the central coordinator) is flexible and adaptive, providing cell admission control and the content of the TDMA frame may change on a per-frame basis to accommodate to actual traffic needs. HiperLan2 also provides power saving and security features as well as mobility support.

HiperLan2 has been designed to carry ATM cells but also IP packets, Firewire packets (IEEE 1394) and digital voice (from cellular phones). The main advantage of HiperLan2 over HiperLan1 is its better quality of service support (low latency) and differentiated quality of service (guarantee of bandwidth), which is what people deploying wireless distribution system want.

The ETSI committee has also specified two additional HiperLan standards with the numbers 3 and 4 called HiperAccess and HiperLink providing outdoor wireless access (with about 5km coverage area between the APs) and high-speed interconnectivity (up to 150m at about 155Mbps) respectively.

3.4.2 HomeRF's SWAP

The HomeRF alliance is a group of big companies from different backgrounds, formed to push the usage of wireless LANs at home and small office (SOHO) area. This group is developing and promoting a new radio LAN standard called SWAP with initial version 1.0 of the standard available since the end of 1998.

The HomeRF task force has decided to tackle the main obstacle preventing wide deployment of Wireless LAN for SOHO use: the cost. Most users just can't afford to spend the money required to buy a couple of radio LAN cards to connect all of their PC equipment without even talking about the access point device. Good 802.11 cards are not really cheap even more than ten years after the first products release, lurking around 50 USD today. The main cost of a wireless LAN card is the radio modem. As this is analogue and high power electronics, it doesn't follows Moore's law so that modems tend to be fairly stable in price. Frequency hopping modems tend to be less expensive but the IEEE 802.11 specification impose tight constraints on the modem (timing and filtering), making it relatively high cost. By releasing slightly those constraints the SWAP specification allows for a much cheaper implementation, but still keeps sufficiently good performance.

The SWAP MAC protocol is implemented in software and digital, so doesn't contribute that much to the final cost of the product. The MAC protocol has been designed to be much simpler than 802.11, combining the best feature of DECT (an ETSI digital cordless phone standard) and IEEE 802.11. The voice service is carried over a classical TDMA protocol (with interference protection, as the band is unlicensed) and reuse the standard DECT architecture and voice CODEC. The data part use a CSMA/CA access mechanism similar to 802.11 (with MAC level retransmission, fragmentation, etc.) to offer a service very similar to Ethernet. Both methods are combined together on the air in a single `SWAP` superframe.

The 1 Mb/s frequency hopping PHY (with optional 2 Mb/s using 4FSK) allows 6 voice connections and enough data throughput for most of the home users and operates in the same ISM band like 802.11 WLANs. The voice quality should be equivalent to DECT in Europe and much better than any current digital phone in the US. End-to-end data performance is expected to be slightly lower than in 802.11 (at equal gross PHY rate). The MAC protocol has been designed to be flexible, allowing to develop very cheap handset or data terminals but also higher performance multimedia cards for PCs if required.

The SWAP specification is an open standard (in fact, more open than 802.11, because there should be no royalty or patent issues), quite simple and straightforward.

3.4.3 Bluetooth

While one may be thinking that Bluetooth [23] is a wireless LAN standard, it rather offers the functionality of a wireless USB (universal serial bus) and was intended as a cable replacement technology mostly developed and promoted by Ericsson with the help of Intel in late 1990s.

Bluetooth delivers the capability to create a set of point to point wireless serial pipes

(RfComm) between a master and up to 6 slaves, providing a protocol (called SDP) to bind those pipes to a specific application or a driver. The Bluetooth mindset is very vertical, with various profiles defining every details from bit level to application level. Bluetooth provides point to point links but no native support for IP. TCP/IP is only one possible profile, implemented through the PPP protocol in a specific pipe. There are other pipes for audio, video, etc.

With Bluetooth nodes need to be explicitly connected, but they may remember bindings from one time to another. This is miles away from the current wireless LAN approach (connectionless broadcast interface, native IP support, cellular deployment, horizontal play), so Bluetooth doesn't fit TCP/IP and wireless LAN applications well. On the other hand, as a kind of wireless USB, it fulfils a role which regular wireless LANs can't, because TCP/IP discovery and binding protocols are too heavyweight.

The Bluetooth system supports both point-to-point and point-to-multipoint connections. In point-to-multipoint mode, the channel is shared among several Bluetooth units. Two or more units sharing the same channel form a Bluetooth piconet. There is one master unit and up to seven active slave units in a piconet (and that is really very similar to USB). These devices can be in any of the following states: active, park, hold and sniff. Multiple piconets with overlapping coverage areas form a Bluetooth scatternet.

A Bluetooth device consists of a radio unit, a link control unit and a support unit for link management and host terminal interface functions. The radio unit operates in the 2.4 GHz ISM band (therefore overlapping regular 802.11b frequencies) on one of the 79 channels (Japan, Spain and France have only 23 channels allocated) of 1MHz each. Depending on the class of the device, a Bluetooth radio can transmit between 100 mW (20 dBm) to a minimum of 1 mW (0dBm) of output power. Bluetooth uses frequency hopping TDD (Time-Division Duplex) PHY for low interference and fading with master-controlled cell clock and transmits using GFSK (Gaussian Frequency Shift Keying) modulation. The nominal hoping frequency is 1600 Hz and determined by a cyclic code of length $2^{27} - 1$.

The Bluetooth MAC layer uses a combination of circuit and packet switching. The channel is slotted and slots can be reserved for synchronous packets. The protocol stack can support one asynchronous connectionless link (ACL) for data and up to three simultaneous synchronous connection-oriented (SCO) links for voice or a combination of asynchronous data and synchronous voice. Each logical voice channel provides a 64Kbps (fitting well the needs of digital telephony standards like DECT or ISDN) synchronous channel in each direction. The asynchronous channel can support a maximum of 723.2Kbps uplink and 57.6Kbps downlink (or vice versa) or a full duplex 433.9Kbps symmetric link. The Bluetooth stack primarily contains a physical level protocol (Baseband) and a link level protocol (LMP) with an adaptation layer (L2CAP) for upper layer protocols to interact with lower layer ones.

Bluetooth packet transmissions are performed in $625\mu s$ time slots with PHY transmission frequency kept constant and with a single packet being transmitted per slot. However, packets are allowed to be transmitted over multiple slots, allowing asymmetric connection speeds. When a packet occupies multiple slots, the transmission frequency stays constant, dropping the hopping rate below 1600 hops per second. The TDD mechanism is implemented by alternating the master and slave transmission slots, with the master transmitting in even-numbered slots, and slave(s) transmitting in odd-numbered slots. In order to maintain this scheme, packets can last one, three, or five time slots.

Main advantage of Bluetooth over 802.11 is its high error tolerance. Bluetooth is intended to properly operate on very low quality channels. While average BER for 802.11 is expected to be about 10^{-6}, it is expected to be as high as 10^{-3} for Bluetooth. In order to reliably operate under such conditions, several layers of error correction have been added to the standard. Bluetooth implements three forms of error recovery: rate 1/3 FEC, rate 2/3 FEC, and an automatic-repeat-request (ARQ) scheme.

An innovative and interesting feature of Bluetooth is the Service Discovery Protocol (SDP). SDP allows a Bluetooth device to query nearby devices and find out which services are available for usage (in analogy to wireline USB).

3.5 Conclusion

Table 3.2: Today's Wireless Technologies

	802.11b/g	**HiperLan2**	**Bluetooth**	**HomeRF**
Operating range	large	large	short	medium
MAC rate	high	high	low	medium
MAC type	CSMA/CA	EYNPMA	TDD	TDMA-CSMA/CA
PHY type	DS/FH	OFDM	GFSK	FH
QoS	no	adaptive	3 voice channels	voice channel

This section provided a short introduction into the IEEE 802.11 standard and related wireless technologies. The broad variety of concurrent wireless technologies permits to make the assumption that the development of WLANs will still continue for years. The development goes obviously into two main directions. The first is to provide high-speed and reliable wireless network technologies intended as replacement for unpractical cable-based Ethernet or as extension of an existing LAN in places, where cabling is either to difficult (e.g. at home or for point-to-point links) or to costly (e.g. due to big distance or requiring earth moving). Second direction for wireless technology development is to provide very low cost, sufficiently fast and quite reliable but short-distance connectivity offering similar functionality as an USB cable in order to provide a standard for device-to-device communication.

The actual 802.11 standard belongs to the first category. seems to be bloated with features and some of them like the PCF access mode are even not well defined (in the sense of the central scheduling algorithm and supported QoS parameters) nor implemented in the majority of actual 802.11 devices. The complexity of each of the functional blocks (PHY, MAC, roaming) in the actual IEEE standard may reduce the compatibility of products from different vendors. Moreover the 802.11 collision avoidance algorithm is still not well understood and subject to various changes and improvements e.g. [134], [60], [100], [31]. Further it is not clear whether and how the retransmissions at the MAC layer in 802.11 may interact with upper layer protocols performing data retransmissions like TCP does.

The future development of 802.11 goes clearly into the direction of improving the gross throughput and the reliability of the PHY but also enhancing the performance of the DCF access method. While the original version of 802.11 provided only a 2 Mbps gross data rate, recent products reach a data rate of up to 108 Mbps (which is not yet an IEEE standard). The advances in microchip technology permit to include more and more powerfull signal processors in even very low cost WLAN devices, so that the reliability of the PHY can be further improved. Recent advances in the 802.11 WLAN technology permit to include a multiple antenna technique called MIMO on low-cost consumer cards, which improves the signal quality in case of fading and heavy interference.

Chapter 4

TCP: Transmission Control Protocol

This section delivers a brief description (may be skipped by readers famillliar with the subject) of the TCP protocol, its basic properties but also its open problems, which are actually subject to ongoing research. TCP is a standardised protocol with IP protocol number 7 (for IPv4 as well as IPv6) with its initial version described by the RFC 793 [68]. TCP provides considerably more facilities for applications than UDP, notably error recovery, flow control and reliability while being connection-oriented, in contrary to UDP, which is an unreliable connectionless datagram delivery protocol. Most of the widely-used user application protocols, like Telnet, FTP or HTTP use TCP to transport user data. Two TCP peers communicate with each other over a TCP connection, which is a virtual data pipe and can be accessed through a standard socket API[133] available across virtually all operating systems and programming platforms. The standardisation and stability of the API over a very long time period (together with its simplicity!) is one major reason for the wide popularity of TCP. Another reason is the great interoperability of different TCP implementations across a wide range of operating systems (e.g. even a very old TCP-Tahoe sender can still communicate with a recent TCP-Vegas receiver).

4.1 Key Features of TCP

The key feature of TCP is the ability to provide reliable logical connection service between two peers over an unreliable datagram transportation protocol like IP. From the application's point of view, TCP transfers a contiguous stream of bytes through the network, however a special option, as stated later, permits transmission of small amount of urgent out-of-band data. The application does not have to deal with chunking the data into smaller blocks, called "segments" from now on. In contrast a "TCP packet" refers to a particular instance of a TCP segment as carried by the physical network. In this notion ACK-only packets are instances of zero-sized TCP segments. The grouping of application bytes into TCP segments is automatically accomplished by sender's TCP stack and then passed to IP (or any lower layer used for transportation) for further transmission. TCP decides itself how to group the application's data and it can forward the segments at its own convenience.

However, sometimes an application needs to ensure that all the data passed to the

TCP layer has actually been transmitted[1] and therefore a TCP `push` function is defined by the standard. It will immediately push all remaining TCP segments still in sender's memory down to the IP layer.

In order to provide reliability, each data byte transmitted by TCP sender has an unique and monotonically increasing sequence number assigned (with the initial sequence number for the first byte in the stream is a random value generated/obtained during TCP connection establishment phase) and must be explicitly acknowledged (`positive ACK scheme`) by the receiving TCP. Therefore a TCP data packet carries only the sequence number of the first data byte in the segment and the sequence number can be called segment sequence number.

If an outstanding data ACK is not received within a timeout interval, the data is retransmitted. The TCP receiver uses the segment sequence numbers to rearrange the segments in memory, when they arrive out of order (e.g. due to non-FIFO network queueing), but also to eliminate potentially duplicated segments[2].

The TCP flow control uses information returned back by TCP receiver with its ACK packets. The receiving TCP when sending an ACK indicates to the sender the additional number of bytes it can still receive (announced `receiver window`) beyond the last received TCP segment, without causing overflow in its internal buffers. This mechanism is also referred to as `sliding window mechanism`.

TCP also provides virtual connections and stream multiplexing achieved through the concept of ports, which are used to augment the receiver and sender address, in order to form a virtual connection identifier (quadruple) of the form {src_addr, src_port, dst_addr, dst_port} uniquely identifying a TCP data stream. Therefore TCP needs to initialise and maintain certain status information for each of its data stream on a particular host. The combination of this status, including OS socket status, individual sequence numbers and window sizes is called a `logical connection`.

4.2 Basic TCP Operation

This section is a reminder about the basic TCP operation based on the `Reno`[3] version of TCP necessary for the development of the VFQ algorithm, however in the next section more details about next generation TCP variants are given.

4.2.1 Connection Establishment and Tear Down

In order to provide reliable delivery of data, two TCP peers must establish a relation with each other before they are able to exchange any application data or in other words the logical TCP connection has to be established between the two endpoints.

[1]a well known example are interactive applications like Telnet, where users tend to produce single keystrokes where any delay would be unacceptable.

[2]the RFC is ambiguous about how the duplicates have to be removed. Recently discovered ambiguities in TCP[121] permit for example OS fingerprinting, as well as firewall tunnelling, etc.

[3]Reno and Newreno TCP have both got their names from the name of the BSD Unix system where they have been implemented first.

One side (usually a server application) performs a `passive open` call, and the other side an `active open` call. The passive connection open request prepares the TCP/IP stack on the issuer side to accept an incoming active open request, which sends the initial `synchronisation packet`. During the connection establishment phase three TCP segments are exchanged:

- packet 1: active open generates a SYN packet with sequence number $seq = n$
- packet 2: receiver side TCP which issued a passive open in advance, generates a SYN-ACK packet with $seq = m$, and ACK $seq = n + 1$
- packet 3: finally the SYN+ACK packet is acknowledged by the sender side with ACK $seq = m + 1$

The logical TCP connection is now established and the two data streams are initialised (the sequence numbers exchanged between the endpoints). This type of connection establishment is known as `three-way handshake`.

Closing the connection is done implicitly by sending a TCP segment with the `FIN` flag set and must be acknowledged by a FIN-ACK packet. Since the connection is full-duplex (that is, there are two independent data streams, one in each direction), the FIN segment only closes the data transfer in one direction. The other side may still send the remaining data it has to transmit and terminate the connection with a TCP segment, where the FIN bit is set. The connection is deleted (kernel level status information on both sides removed) once the data stream is closed in both directions, however a special cookie may stay in the operating system kernel that prevents application from reusing the same endpoint for a short time period (TIME_WAIT state). Closing a TCP connection in one direction forces a buffer flush (push) of the queued data in that direction.

4.2.2 Sliding Window

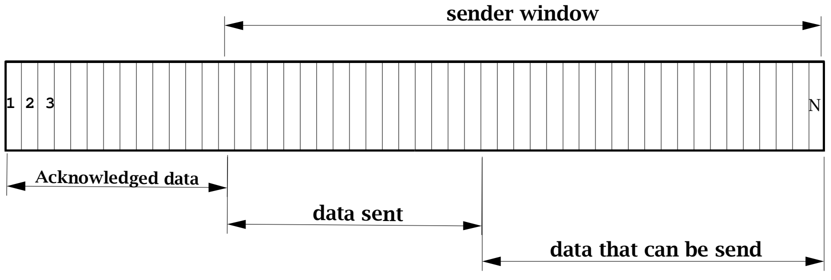

Figure 4.1: The TCP sliding window principle.

A reliable transport protocol must use an acknowledgement scheme in order to obtain information about packet arrival at the receiver side. The simplest possibility would be a send-and-wait scheme: send a packet and then wait for an acknowledgement from the receiver before sending the next packet. If the ACK is not received within a certain amount of time, retransmit the packet and so on.

While this simple mechanism would already ensure reliability, the sender would not be able to fully utilise available network bandwidth. Since an average network path can be considered as a pipeline, where each network segment (hop) is able to buffer some amount of packets[4], it is advantageous to have multiple packets in transit. Therefore network protocols tend to use a window based approach, where the sender is allowed to send more than one packet at a time from a variable size window of ready-to-send data. A `congestion control algorithm` is necessary to properly estimate the optimal size of the permitted sender window, called congestion window (`CWND`) for that reason. A corresponding receiver side sliding window `RWND` is also necessary to cope with segment reordering or missing packets and its size must be signaled to the sender side. Finally the effective window used by the sender (that is the maxiumum of data bytes in transit) is set to $min(CWND, RWND)$.

TCP uses the above window principle with some modifications: since TCP provides a byte-stream connection, sequence numbers are assigned to each byte in the stream. TCP divides this contiguous byte stream into appropriate TCP segments of variable size, in order to optimally fit the maximum packet size of the underlying transport protocol (usually measured by path MTU discovery). The window principle is used at both sides of a full-duplex TCP connection with byte granularity, which means that segments sent and ACKs received carry byte-sequence numbers and the window size[5] is expressed as a number of bytes, rather than a number of packets[6].

The size of the initial sender window is set to a fixed value (depending on the particular TCP implementation and the TCP algorithm variant)[7] and the size of available buffer space on the receiver side (`RWND`) is announced, when the connection is established. Both windows are variable while the data transfer proceeds: the sender window will follow the value obtained from the congestion control algorithm, while the receiver window will be limited by the available buffer space (in practice a "slow" application sending TCP data may bound the size of the effective sender window). The available receiver buffer space can be modified by the application through the socket API (but is usually limited to a maximum value by the OS kernel) on a per-connection basis. Each TCP ACK segment will also include the actual receiver window size for flow control purposes. The relation between the described TCP window variables is shown in the figure 4.1.

4.2.3 Acknowledgements and Retransmissions

TCP acknowledgements specify the sequence number of the first byte the receiver side expects to receive as next. If a data segment from a TCP stream is lost in transit, the receiver will still acknowledge all further well-received segments but using an ACK sequence number referring to the first byte of the missing segment. If the sender window is bigger than just one segmnet, the loss may[8] generate duplicated

[4]this includes queue buffers but also the inherent link delay. For example a transatlantic optical link can have multiple packets "on wire", since speed of light is limited.

[5]however if the window scaling option described later is used, the window size may be expressed in multiple of bytes

[6]in opposite to common simulation software like NS2, where a simplified packet-only approach has been implemented so far. Special care must be taken while compareing NS2 simulation results with real TCP implementations.

[7]however the RFC [68] states, that any value between 1 and 4 MTU segments is valid.

[8]if the duplicated ACK packets are not lost in transit on the reverse path too.

acknowledgements (DUPACK) that arrive at the sender. Otherwise a retransmission timer may expire and cause a retransmission of the next outstanding segment. Since the sender will stop transmitting after it has sent all the bytes permitted by its actuall sending window, the retransmission timer will also be used to trigger the retransmission of an outstanding segment, if no ACK packets arrive at all (e.g. due to very heavy congestion on the reverse path) prior to its expiration. The timeout value is calculated taking the round-trip time of the network path into account. This may lead to problems, if the RTT is either not stable or the TCP sender could not calculate a good estimation for this value yet, like at the very beginning of a new connection.

If a retransmission situation is detected by the sender (either by the DUPACK mechanism or a timeout), the sender side must decide, which packet requires retransmission - this is of course at least the first unacknowledged segment, but the sender does not have any knowledge what happened to the following segments (a selective acknowledgement modification to TCP (SACK-TCP) discussed later has been proposed and is available in state of the art OS software like Linux). The TCP sender may decide to retransmit the first missing segment only but also some more segments from the remaining part of the sender window. However, most of the existing real-world TCP implementations will retransmit the next segment only.

In order to correctly operate under any network conditions, a TCP-compliant sender should implement an algorithm to adapt the retransmission timer timeout values to the round trip time of data packets through the network. This is accomplished by recording the time, at which a packet is sent and the time at which the corresponding ACK is received. A moving weighted average (`EWMA`) is calculated over several of these round trip times and used as timeout base value[9] for future segments sent.

An important feature of TCP acknowledgements is their cumulative character, that is an ACK with sequence number set to N[10] tells the TCP sender, that all of the outstanding data segments with sequence number of the last byte less than N have left the network and have been accepted by the receiver. Therefore a TCP connection is much less sensitive to packet losses from the ACK stream than from the data stream with respect to the achievable throughput. Moreover, each lost data segment always means wasted network bandwidth, since it will be retransmitted later, while some lost ACKs may even improve the total connection throughput, since more bandwidth may become available for data packets (see later).

4.3 TCP Packet Format

TCP is a quite complex protocol and for this reason the TCP header as shown in the figure 4.2 includes numerous fields to carry out different protocol functions, which are described now in more detail. All TCP header fields spanning more than a single byte are always given in the network byte order[11].

[9] The RFC [68] states that the retransmission interval should be set to 4 times the path RTT.

[10] Attention to sequence number wrap around, which is an issue in very high speed networks like 10GB ethernet.

[11] which means "MSB first".

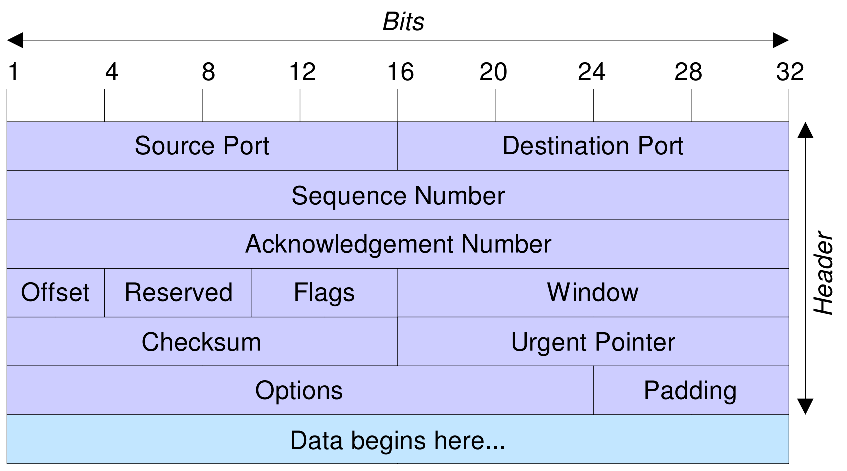

Figure 4.2: TCP header format.

The `source port` and `destination port` number fields specify the virtual connection endpoints the TCP connection belongs to and are both 16 bits in size. The two adjacent sequence and acknowledgement number fields give the sequence number of the first byte carried by the TCP segment and the next byte sequence number expected by the receiver side, respectively. They are interpreted in context of the setting of the `flags` control field that contains 8 function bits.

The `flags` field selects the TCP function to be performed for a given TCP packet and consists of 6 bits named `FIN` (bit 0), `SYN`, `RST`, `PSH`, `ACK` and `URG` (bit 5) respectively. The remaining two highest order bits are marked as reserved in first versions of TCP RFCs, but recently they have been reassigned to the `ECN` (for Early Congestion Notification) extension of the TCP congestion control discussed later.

If the `SYN` control bit is set, the `sequence` number field carries the initial sequence number N for the connection and the first data byte in the TCP stream is assigned the sequence number $N + 1$. This is only used during the connection establishment phase.

If the `ACK` control bit is set, the `acknowledgement` field contains the value of the next sequence number that the receiver is expecting to receive. Available TCP implementations like the one in the Linux kernel, always transmit a valid ACK sequence number with all data packets.

The `FIN` control bit indicates (if set), that the sender of the packet terminated his half of the full duplex connection. The `RST` bit immediately resets the connection and may be set in a response TCP packet, if a station receives any TCP data where no connection state exists in the station TCP stack. This may happen in the case of forged (malicious) TCP packets or if a machine suddenly reboots[12] but the other side of a previously established connection sends a packet. A TCP sender is permitted to accept a `TCP RST` packet, only if the sequence number of the transmitted segment

[12]**Some very popular OS software is well known to exhibit such "sudden reboot" property.**

matches its actual sliding window for the connection. Otherwise an eavesdropper could disrupt a TCP connection by guessing only connection's port and IP numbers without regard to the actual sequence space and would make a denial-of-service attack rather easy.

The `PSH` bit implements the `PUSH` function described above and `URG` bit is used to deliver small portions of out-of-band data and indicates, that the `urgent pointer` field is valid and points to the byte of urgent data in the segment body. Urgent data is only delivered upon a separate request from the application and is used for example by Telnet for session interrupt purposes.

The `offset` field indicates where application data starts in the segment body and is necessary since the TCP header is of variable length due to possible additional options, which may be included at the end of the regular header. The two byte `window` field carries the size of receiver window, that is the actual amount of buffer space available at the endpoint (the amount of bytes the receiver can still accept), at the time the segment has been sent. Finally, the `checksum` is a 16-bit one's complement of the one's complement sum of all 16-bit words in the so called `pseudo-header` (built of the source and destination IP address from the two connection endpoints), the TCP header and the TCP data. While computing the checksum the checksum field itself is set to all zeros.

4.4 TCP Congestion Control and Avoidance

As a general-purpose transport protocol TCP must be able to operate under any network condition and therefore requires a network capacity estimation algorithm in order to adapt to a broad variety of network bandwidths and types. Despite its age, TCP can still be the choice for backup and other applications requireing bulk connections, where huge end-to-end throughput is the primary concern (one example is the CERN Data GRID project, where enormous amount of experimental data must be transferred between the data detectors and the end storage with actual capacity estimation of roughly 20 gigabytes per second [30]), but delay and jitter are usually of less importance. This situation is contrary to the needs of an average home use of the Internet, where a relatively slow (in respect to e.g. GRID requirements) DSL line (or even worse: an analogue modem connection) is used for web browsing and media streaming and requires good response times for short living connections but also stable delay.

Several congestion control enhancements have been added to and suggested for TCP over the years and this is still a very active ongoing research area. However, most of the state of the art TCP implementations (like those present in consumer operating systems including latest Linux and MS Windows versions) provide a so called Newreno type TCP with slow start, congestion avoidance, fast retransmit and fast recovery as its key features.

Depending on the particular application scenario, classical Newreno TCP may exhibit a number of undesirable properties which can be attributed to its rather unflexible congestion control and avoidance algorithm. The relevant problems of TCP with respect to a wireless environment will be studied later in this chapter. Note that for each particular application scenario, TCP congestion control and avoidance can yield a different short- and long-term behaviour.

4.4.1 Slow Start

While the initial TCP specification permitted TCP senders to inject multiple segments into the network up to the window size advertised by the receiver (which may be acceptable on a LAN but if the link is very slow, the initial burst of packets may already cause network congestion), later refinements of the TCP specification mandated a `slow start` phase.

During the slow start[13] a TCP source operates by assuming that the rate at which new packets should be injected into the network is the rate at which the acknowledgements are returned by the other endpoint. If in slow start, sender's CWND is effectively doubled after each CWND of data sent (therefore it is rather an exponential start). This is accomplished by increasing the congestion window size each time an ACK packet is received by one segment. At some point of this process the available path capacity may be reached and an intermediate router will start to discard packets. This will tell the sender (by DUP-ACK or timeout) that its congestion window has became too large.

While this scheme works well for an average persistent TCP connection, it fails to address some of subtle problems, if different types of connections have to cross a single bottleneck. New TCP connections are penalised ([130]), since the peers did not yet obtain correct information about the path RTT and the available network bandwidth. Also if a standard TCP receiver have to compete with a `DelAck`[14] receiver, the second one is penalised, since a `DelAck` receiver will send an ACK packet only after every other data packet it receives (or after a timeout occurs, however during connection establishment TCP timeouts tend to be too large). Therefore a `DelAck` receiver will permit its peer to increase the CWND only by a factor of 1.5 after each window of data sent.

Recently the `byte counting` [5], [4] technique has received increased attention in the research community. Byte counting is a modification to the default acknowledgement policy of TCP permitting the receiver to generate one ACK packet for every fixed amount (algorithm's tunable parameter) of bytes received instead of generating one ACK for every (or every second for DelAck) new data segment, making the expansion of the sender congestion window independent of the details of data packetisation (e.g. with normal TCP two sessions with equal RTT values but a 1000 versus 1500 byte MTUs would have expanded their CWND to 8000, respectively 12000 bytes after 3 RTT cycles while in slow start).

[13]slow start means just a start from a (s)low sending rate, but not that it is done slowly by itself!

[14]Most of actual TCP/IP stacks use the `DelAck` approach by default, however for example under Linux, it is a configurable feature.

4.4.2 Congestion Avoidance

The slow start phase ends if packet loss is detected by the TCP sender and the congestion avoidance phase begins. The assumption of the congestion detection algorithm is that packet loss caused by damage[15] can be neglected. Therefore the loss of a packet signals congestion somewhere in the network between the source and destination.

Congestion avoidance and slow start are independent algorithms, however in practice they are implemented together. Congestion avoidance and slow start require that two variables are maintained for each TCP connection: a congestion window `CWND` and a slow start threshold `ssthresh` that are initialised to one segment and `CWND_MAX` (that is 2^{16} bytes if window scaling is not used) respectively.

Upon a detected congestion event one-half of the current congestion window size is saved in the `ssthresh` variable and if assumed congestion was due to a timeout, `CWND` is reset to one segment (more precisely: to the initial value). With each ACK that is successfully received, `CWND` is increased depending on the operation mode: if `CWND` is less than or equal to `ssthresh`, TCP is in slow start mode, otherwise TCP is performing congestion avoidance.

If in congestion avoidance, `CWND` is incremented by one segment after the current window of data has been sent, resulting in a linear growth of `CWND`[16]. For an unknown reason (probably an implementation error) some real-world TCP implementations incorrectly add a small fraction of the segment size (typically the segment size divided by 8) during congestion avoidance to `CWND` value.

4.4.3 Fast Retransmit

`Fast retransmit` is a modification to the initial TCP congestion avoidance algorithm proposed in 1990 (see RFC [122]) and uses the DUP-ACK mechanism to recognise a moderate congestion situation, while still keeping the timeout mechanism for severe congestion situations. The idea is simple: if more than a predefined number of duplicate acknowledgements arrive, the sender assumes that the congestion is moderate (since the acknowledgements still arrive, which means that corresponding packets have not been dropped) and resends the missing segment without entering the slow start phase and cutting the congestion window to one segment. Since TCP does not know whether a duplicate ACK is caused by a lost segment or just by a reordering of segments in the network[17], it waits for a small number (configurable parameter more than 2, usually 3, called `dupack threshold`) of duplicate ACKs to arrive.

While being effective for bulk connections, `fast retransmit` fails to improve TCP operation, if the current sender congestion window is very small. The window on

[15]**which means packet drops due to incorrect checksum or just packet loss on the PHY medium.**

[16]**Linear with respect to the number of data windows sent. The exact CWND evolution over time for constant RTT value can be described by a square-root curve between two congestion events. This follows from the corresponding differential equation for the convolution of the $CWND(t)$ curve at the points where the window size is increased.**

[17]**non-FIFO scheduling elements in the network may do this.**

the sending side may be too small in order to generate enough duplicate acknowledgements to hit the DUP-ACK limit and the sender may not be able to enter the fast retransmit phase after a loss of a segment. To mitigate this problem a `dynamic fast retransmit` scheme has been proposed in the publication [124] and evaluated in the context of wireless network application (here wireless network usually means "low-speed" and "moderate delay", that is rather low average CWND value. High-speed networks like 1GB or 10GB ethernet usually do not suffer from this particular problem).

4.4.4 Fast Recovery

`Fast recovery` is the ability of the sender TCP's to enter congestion avoidance instead of slow start, after the fast retransmit algorithm sends what appears to be the missing segment. It is an improvement which allows much higher throughput under moderate congestion especially for large sender windows compared to older TCP.

A special algorithm called temporary window inflation is used during fast recovery to account for segments which have already left the network and have triggered duplicate ACKs. When the third (or according to the dupack threshold value) duplicate ACK in a row is received[18], `ssthresh` is set to one-half the current congestion window but no less than two segments and the missing segment is retransmitted. Then the `CWND` is set to `ssthresh` plus 3 times the segment size in order to account for the three duplicate ACKS. Each time another duplicate ACK arrives, `CWND` is incremented by one segment size. This inflates the temporary congestion window for each additional segment leaving the network and then next packet is transmitted if permitted by the new value of `CWND`.

If a higher (that is acknowledging new data) ACK arrives now, `CWND` is set to the value of `ssthresh` in order to undo the temporary window inflation. In a moderate congestion situation the new ACK should be the acknowledgement for the missing packet and received roughly one round-trip time after the retransmission. Additionally, the new ACK may acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicated ACK packet.

4.5 Next Generation TCP Algorithms

The preceeding section delivered an overview of the classical TCP and its operation focusing on the Reno version. Newreno TCP is just a minor modification to Reno's fast recovery mechanism that improves throughput by avoiding multiple reductions of the congestion window, if more than one packet is dropped from the same window of data [45] therefore further improving throughput in a moderate congestion situation. While Newreno TCP was state of the art for long time, huge amount of research in the area of TCP behaviour under non-optimal network conditions, TCP congestion control as well as congestion avoidance, appeared from 1990 on.

Many of the issues related to TCP performance over wireless type links (see for ex-

[18]**Until dupack threshold is reached, new packets are clocked out on duplicate ACKs, if permitted by the effective (inflated) sender window.**

ample [126]) are aimed by those publications, but some are not and seem to receive increased researcher attention during the last two years. TCP problems related to wireless link errors, which have gained fairly most attention in the literature, will be briefly sketched in the next chapter. A short look at two prominent general-purpose developments and their relation to Reno TCP is given below, followed by a short description of an alternative congestion avoidance scheme called ECN-TCP.

The practical evaluation of TCP is a challenging task since the protocol is so versatile and complex. A systematic introduction into understanding of the TCP universe can be found in the publication [6].

4.5.1 TCP Vegas

TCP Vegas is a sender side modification of the Reno segment transmission scheme (directly interoperable with any other TCP receiver) and has been introduced in the two papers [25] and [26]. Vegas TCP approaches the problem of network congestion from a different perspective than Reno. A Vegas TCP sender calculates the currently available network bandwidth by continuously observing the packet RTT and dividing the size of the actual congestion window by this value. This `observed rate` is compared to so called `expected rate` based on a base-RTT estimate, that is the up-to-date best RTT (numerically lowest) value experienced in recent past. Two configurable parameters α, β are then used to perform a dynamic control algorithm for setting the sending rate using the gradient[19] between the two bandwidth estimates, indicating that the channel is either under- or over-utilised. Vegas-TCP also uses the fast retransmit and fast recovery mechanisms known from classical Reno, but has a different slow start and congestion avoidance behaviour.

The novel idea of TCP Vegas is therefore the attempt to estimate and recognise any path congestion before it actually happens trying to avoid unnecessary congestion and packet losses at all. In contrary, Reno induces congestion in order to estimate network capacity, since it uses packet losses as congestion indicator. A disadvantage of Vegas over Reno is its higher susceptibility to congestion on the reverse (that is ACK carrying) path, which has also an impact on the total observed RTT and therefore on the bandwidth estimates used by Vegas.

In the `slow start` phase Vegas doubles its actual `CWND` value after every other window of data sent (this is similar to Reno-TCP working together with a SACK-TCP on the receiver side described later) in opposite to Reno doubling `CWND` after each window of data sent. The slow start phase ends, once Vegas detects queue build up, which is assumed to happen if the difference between the `expected rate` and the `actual rate` exceeds a certain threshold defined by $\frac{\gamma}{BaseRTT}$. This more inteligent approach helps to avoid losses during the slow start phase but ends usually with a maximum `CWND` far less than in a Reno-like slow-start.

During the `congestion avoidance` phase, Vegas decides if it should increase or decrease (or even keep constant) its actual `CWND` value by one segment after each

[19]**Original Vegas TCP does not utilise the absolute value of the difference between the two values, but only the sign to decide whether to lower or increase the rate.**

window of data sent, in contrast to Reno which will always increase `CWND` during congestion avoidance. The estimation of the current channel condition is performed by calculating the expected channel capacity as $T_{exp} = \frac{M}{BaseRTT}$, where M is the number of bytes in transit (unacknowledged and retransmitted segments in fly) and comparing it to actual bandwidth obtained from a probe packet and the corresponding RTT as $T_{act} = \frac{N}{BaseRTT}$, where N stands for the number of bytes sent between the probe packet and the arrival of corresponding ACK packet.

Based on the two above estimates, a dynamic congestion control is applied to the sender side window size `CWND` as follows. If the relation holds:

$$T_{exp} - T_{act} \leq \frac{\alpha}{BaseRTT} \tag{4.1}$$

then `CWND` is increased analogically to Reno by one segment, otherwise if

$$T_{exp} - T_{act} > \frac{\beta}{BaseRTT} \tag{4.2}$$

then Vegas' `CWND` is decreased by one segment after each window of data sent. Between the two thresholds, the `CWND` size is kept constant providing a steady-state region. Common setting for the parameters is $\alpha = 2, \beta = 4, \gamma = 1$ and called Vegas(2,4)[20].

Vegas' fast retransmit and fast recovery algorithms are similar to Reno, however the packet loss detection algorithm has been improved. While Reno normally waits for three duplicate ACKs to recognise moderate network congestion, Vegas-TCP is able to enter fast retransmit even after just a single duplicated ACK under the assumption, that the first duplicated ACK comes from the packet following the first unacknowledged packet (that is packet reordering does not take place) and compareing the round trip value calculated from the duplicated ACK with a timeout value assigned to the first unacknowledged packet. Similar timeout check is performed for the first and the second packet sent by Vegas after a retransmission on reception of a new ACK to determine if they may have been lost.

TCP Vegas has been shown ([3], [36], [95], [24], [51]) to perform less retransmissions than Reno under the same network conditions (that is for the same packet loss and delay), therefore providing better utilisation of the available network capacity. The absolute throughput gain with respect to Reno sending over an equal network path has been shown [26] to be between 37% and 71% while performing only one-fifth to one-half as many retransmissions. In a direct competition with Reno sharing a common bottleneck path, Vegas connections are known to obtain less throughput than Reno flows. Depending on the given scenario, Reno-TCP can outperform Vegas-TCP by a factor of 3 with respect to throughput.

It is also known that Vegas' stability and fairness improve with the increasing buffer capacity along its transmission path, while Reno TCP is much less sensitive to the available buffer space [3]. This is logical, since Vegas TCP requires some amount of segments to be queued at a bottleneck link in order to detect increasing path RTT at the moment congestion starts to build up. On the other hand Reno-TCP is

[20]**NS2 default setting is** $\alpha = 1, \beta = 3, \gamma = 1$.

only sensitive to bottleneck queue overflows, so that it can reliably detect congestion regardless of the absolute amount of network buffers available along the path. An inmportant advantage (e.g. for multimedia applications) of Vegas TCP is its smoother sending rate compared to the average (that is very bursty) Reno behaviour.

The apparent failure of Vegas' novel congestion avoidance scheme in direct competition with Reno (see also later in the simulation section) in respect to resulting bandwidth sharing can be attributed to its conservative nature and huge stability area, where its congestion window size may remain constant [95]. Once a Vegas source has estimated the actual network delay, it may ignore additional bandwidth available in the network and reach steady state too early, resulting in an unfair bandwidth distribution between the sources. A more aggressive setting for the Vegas control parameters of $\alpha = 2, \beta = 2$ is analysed in the Vegas evaluation [95] and found to yield better bandwidth fairness, while as a price for this increase turning Vegas into a Reno-like algorithm with periodic `CWND` oscillations and possible bottleneck queue overflows. Similar oscillatory behaviour of Vegas for certain settings of α, β has been theoretically derived in the fluid approximation model for TCP discussed in the paper [24].

In an arbitrary network topology it is not easily possible to guess which behaviour (aggressive-oscillatory a la Reno or defensive a la Vegas) is the right one, in order to deliver fair bandwidth sharing between the concurrent TCP flows. A short example provides an insight into the core of the problem: one extreme topology is a single highly congested network link. In such scenario fair bandwidth sharing between TCP sessions depends on their ability to obtain (on the average) fair shares of the bottleneck buffer capacity, that is, to on average store equal amount of bytes in the bottleneck queue. The right strategy in this case would be to aggressively send a window of data on the network, trying to catch some buffers in the queue. Vegas congestion control however, tries the opposite thing and lowers its sending rate once congestion is detected. On the other hand, in a topology with multiple congested links where some kind of `SFQ` (Stochastic Fair Queuing [94]) policy is deployed in bottleneck routers permitting a flow[21] to always store for example one packet at the bottleneck, the right policy would be to pace (delay) the sending of outstanding TCP segments over some interval, in order to permit the network path to correctly pipeline the packets. Here an aggressive sending of data would lead to very bad performance, since most of the packets would be dropped at the first congested router. This consideration shows, why it is so difficult to find a well-behaving congestion avoidance algorithm, that on the other hand always delivers reasonable performance and fairness.

Despite above limitations, the advantage of the defensive Vegas behaviour is that it does not induce queue oscillations, therefore providing much more stable delay and jitter to the end hosts. However, as pointed out in the paper [26] there is a risk, that a wide-scale (e.g. in consumer TCP/IP stacks like in Windows) deployment of Vegas-TCP could surprisingly increase the permanent congestion in the Internet, since buffer occupations of multiple Vegas flows at bottleneck links may add up and

[21] **Under classical SFQ policy a flow means an aggregate of real flows which are hashed into a single bucket.**

permanently increase round trip times, while with Reno's oscillating queues there are always phases where the queue length is less than the average, permitting a fraction of packets to pass with significantly less delay.

A comprehensive study of the novel techniques introduced in Vegas-TCP can be found in the TCP evaluation [57]. The conclusion is rather disillusioning and contributes the throughput improvements[22] not to the novel congestion avoidance scheme of Vegas, but rather to the modifications to the slow-start and its retransmission behaviour. Authors report that the congestion sensitive slow-start algorithm of Vegas is much more successful in avoiding timeouts during the initial connection establishment phase, while Reno's faster and unresponsive exponential opening of the congestion window may more frequently result in overshooting of the available bandwidth and loosing multiple segments from a single window of data very early, leading to continuous near slow-start operation of Reno sessions and unacceptable performance.

The publication [57] shows that the changes to the segment recovery algorithm of Vegas have the biggest impact on the end-to-end performance. The ability of Vegas to retransmit packets before three DUP ACKs arrive helps to avoid multiple drops from a single window of data.

4.5.2 TCP Westwood

TCP Westwood (TCPW, [29]) is a sender-side modification of the Reno congestion window control algorithm and claims to improve upon the performance of TCP Reno in wired as well as in wireless networks. The key idea of TCPW is to continuously measure at the TCP sender side the bandwidth obtained by the connection via monitoring the rate of the returning ACK packets. The bandwidth estimate is then used to dynamically compute congestion window size and slow start threshold after a congestion event (rather than to unflexibely cut the CWND size to half like Reno does). In contrast to Vegas, which operates on rate variables, TCPW operates on window variables like Reno. The bandwidth estimates exploited by Vegas and Westwood differ substantially from each other: while Vegas uses sender side bandwidth estimate (look ahead), TCPW utilises the receiver side bandwidth (look back, closed loop) being therefore more accurate.

The novel feature of TCPW is called `faster recovery` and is a modification to the fast recovery mechanism of Reno. The TCPW sender monitors the ACK reception rate together with information carried by ACK sequence numbers to estimate the available bandwidth in the connection's forward path. For each pair of consecutive ACK packets received by the sender, a sample bandwidth estimate is calculated by dividing the amount of data delivered between the two ACKS by the elapsed time. Then a low-pass filter[23] is applied to the sequence of bandwidth estimates, in order to obtain the so called low-frequency component `BWE` of the available network bandwidth. TCPW makes therefore the assumption, that under moderate network

[22] In case of a Vegas-only network.

[23] This is usually an EWMA-like discrete filter, however the NS2 implementation of TCPW available at the time of this writting has 10 differen filters. Depending on the filter used, TCPW may exhibit different behaviour. Therefore it is legal to talk about the TCPW family of algorithms.

congestion (which is usually assumed for the fast recovery phase) the source of packet losses are short peaks of excess traffic that periodically reduce a persistent long-living available bandwidth. This is a plausible assumption taking into account that the majority of deployed TCP implementations are Reno-like bursty ones.

If a packet loss is detected during the congestion avoidance phase (by same mechanisms as in Reno), TCPW sets the congestion window and slow start threshold to the value obtained from the filtered bandwidth estimate and the minimum RTT as $BWE \cdot RTT_{min}$ and then performs a Reno-like fast retransmit and recovery procedure.

A more sophisticated modification to the original Westwood algorithm called Westwood+ has been proposed in another article [50], in order to address problems of original Westwood TCP in presence of ACK compression. ACK compression is a phenomenon which may occur in real networks, if ACK segments are reordered by intermediate nodes [96] leading to modified RTT times at the sender. Since Westwood's bandwidth estimate depends on the temporal spacing and the jitter of returning ACK segments, it will fail to correctly estimate available bandwidth, if adjacent ACK segments are significantly reordered or delayed.

TCPW congestion control has been shown [51] to be TCP-friendly, that is to well coexist with previous older TCP versions sharing same bottleneck and to exhibit TCP fairness property at least at the level of Reno. TCPW is also less biased towards the path's RTT values than Reno-TCP. While Reno's steady state throughput is proportional to $1/RTT$, TCPW exhibits only a $\sqrt{1/RTT}$ dependency on the path's round trip time.

Westwood's adaptive window decrease mechanism improves the stability with respect to standard TCP window control and provides sufficient shrinking of the congestion window in the presence of heavy network congestion, but on the other hand shrinking the CWND not too much in presence of light congestion only or in presence of packet losses, which are not related to congestion at all (for example due to wireless transmission errors).

A disadvantage of TCPW is its aggressive Reno like bandwidth probing during the congestion avoidance phase. While Vegas, at least in theory, can keep its congestion window constant during the CA phase, Westwood and Reno TCP will periodically overflow network queues and detect the maximum path capacity only via the duplicated ACK mechanism.

4.5.3 ECN-TCP: Explicit Congestion Notification

Congestion avoidance and detection has been believed to be an end-to-end issue for long time, but could only be solved insufficiently with existing TCP algorithms. Therefore recent research is going into another direction trying to resolve the congestion problem locally, that is at the place where it happens: in intermediate nodes with bottleneck queues. A proposal has been made to add `ECN` into the IP core networks [113] and incorporate a cooperative feedback algorithm into TCP senders and receivers.

The idea of ECN is to use direct congestion notification from intermediary nodes instead of the classical end-to-end approach for CWND controlling. ECN requires both TCP sender and TCP receiver to be ECN-capable and the intermediary network to support ECN signalling at least in some of the intermediary nodes.

A TCP-ECN connection requires a special ECN-handshake during the connection establishment and modifies the initial segment exchange to include ECN capability information. The two highest previously reserved TCP flag bits are used for this purpose. Once established, an ECN connection continues normally until information about congestion building up arrives at the receiver side. Congestion signaling is performed by ECN-capable intermediary nodes by setting the ECN-field in the TCP header to `congestion experienced` bit value. The receiver echoes the ECN information to the TCP sender together with its ACK packets, until a packet with `CWND reduced` setting in the ECN bit field arrives.

By performing this closed-loop algorithm the sender side can estimate network capacity with more accuracy than if TCP had to overflow the queue and receive duplicated ACK packets, given that the intermediary nodes provide ECN information sufficiently early, that is before the full buffer space at the bottleneck is occupied. An ECN-mode is usually included in nodes implementing the RED[24] queue policy ([46],[44]) and modifies the classical RED algorithm to mark TCP packets in the ECN bitfield instead of dropping them, if the bottleneck queue starts to build up.

A practical problem has been reported to exist in today's Internet [103] which reduces interoperability of ECN with non-ECN hosts and is related to the use of reserved TCP bits. Due to oversimplistic or non-ECN capable packet filters, ECN-connections may fail to pass firewall hosts. For same reason there are also issues with interoperability of ECN-aware and non-aware operating systems. In the study [103] authors report that about 6.6% of Internet web servers do not respond to ECN-enabled TCP handshake at all, while 87.5% to ignore the ECN bits. Finally, less than 1% of tested machines have been found to be ECN-compliant.

A comprehensive evaluation of TCP-ECN can be found in the publications [108],[109], where TCP-ECN performance has been measured in the context of large file transfers and compared to standard TCP under Drop Tail and RED policy at the bottleneck. A further improvement to ECN is proposed in the article [90]. The ECN extension to TCP is believed to convey congestion information in a timely manner and diminish packet drops thus increasing the delivered-to-dropped packet ratio. Combined together with configurable RED gateways it is also possible to provide a mechanism for TCP QoS differentiation which enables network operators to throttle throughput of TCP sessions in a non-intrusive way (as it would be if packets would be just dropped) and also permits for example to charge customers for their contribution to the network congestion.

[24]**For Random Early Detection in case of ECN or Drop in case of non-ECN mode.**

4.6 Other Next-Generation Protocols

After providing a deeper discussion about TCP and its properties, I want to complete this section by giving a short overview of TCP-like protocols currently being worked on.

4.6.1 SCTP: Stream Control Transport Protocol

The Stream Control Transmission Protocol (SCTP, see [123], [102]) is a new IP transport protocol, existing at an equivalent level as UDP and TCP[25] and has been designed by the IETF SIGTRAN working group, which has released the SCTP standard draft document in October 2000. It is a general-purpose protocol like TCP and aimed at message-oriented applications well fitting needs for transportation of signalling data.

SCTP provides a number of additional features over TCP which are considered critical for signalling transport, while offering increased performance and reliability. SCTP can be used as the transport protocol for applications, where session monitoring and detection of connectivity loss are required. For such applications, some advanced failure detection mechanisms have been implemented. The core original features of SCTP are multi-streaming and multi-homing while keeping a similar socket API like TCP and UDP[26].

While a TCP data stream is referred to as a sequence of bytes, a SCTP stream represents a sequence of messages (which may be of arbitrary length). The multi-streaming feature allows data to be partitioned into multiple streams which have the property of being delivered independently. Therefore a message loss in any of the streams will only affect delivery within that stream, but not in the other streams. In contrast, TCP provides a single stream of data and ensures that delivery of that stream takes place with perfect sequence preservation, but it may stall for an unacceptable long time, when a single message loss occurs, especially if the recovery happens through the timeout mechanism.

SCTP's multi-homing feature is the ability for a single SCTP endpoint to support multiple IP addresses. Using multi-homed SCTP, redundant paths (using a different route for each stream) can be used for improved fault tolerance. In its current form, SCTP does not do any load-sharing over the assigned addresses, that is, multi-homing is used for redundancy purposes only. The multi-homing feature of SCTP has been subject of the thesis [55].

SCTP's congestion control ([137]) is similar to TCP and uses a window based approach therefore exhibiting similar fairness properties as TCP. SCTP is believed to be TCP-friendly. There is also ongoing research in the area of multipath congestion control trying to incorporate improved congestion control into SCTP using ideas which were already investigated for TCP (selective acknowledgements, Vegas-like CC, etc.).

[25]since the 2.6 Linux kernel version an implementation is available.
[26]the user just needs to specify SCTP as socket protocol.

4.6.2 UDT: UDP Based Data Transfer

UDT [54] is a full-duplex application level protocol, which originated from the SABUL [53] algorithm, introducing new reliability and congestion control mechanisms into UDP for bulk data transfers. An UDT endpoint consists of two parts: the UDT sender and the UDT receiver. The sender sends and retransmits data packets according to a flow- and congestion-control mechanisms, but also additional control packets. The UDT sink receives both data and control packets, processes them properly and may also generate control packets for feedback.

UDT uses packet sequence numbers and ACK sequence numbers with sub-sequencing. Each UDT data packet is assigned an unique and increasing packet sequence number ranging from 0 to $2^{31} - 1$. Each ACK is also assigned an unique ACK sequence number (independent of the packet sequence space) ranging from 0 to $2^{16} - 1$. The receiver side uses an ACK timer to generate periodical selective acknowledgements with a time base of $0.01s$. A separate negative acknowledgement (NACK) timer is used to generate loss reports, if retransmissions are not received within an increasing time interval with the base time value equal to the path RTT.

In opposite to TCP, UDT uses an acknowledgement-of-acknowledgement scheme. The UDT sender will send an ACK2 packet immediately after it receives an ACK packet. The UDT receiver can then estimate the actuall RTT value according to the timestamps when the ACK leaves and when the ACK2 arrives. The receiver side may also generate an ACK2 packet, immediately after it receives a duplicated ACK with the same ACK sequence number as a previous one. The peer side can then avoid generating unnecessary ACKs and the ACK/ACK2 pair can be also used to calculate RTT using the `packet pair algorithm` [80].

The UDT protocol uses rate-based congestion algorithm and window based flow control. Rate control tunes the inter-packet spacing every constant time interval, which is set to 0.01 seconds. This is slightly similar to `TCP pacing` studied in the papers [2], [10] (however the pacing algorithm does not seem to deliver substantial improvements in case of TCP according to those publications). UDT flow control method uses a window approach, in order to limit the number of unacknowledged packets in the network. Every 16 data packets, the sender sends out the next data packet immediately, without regard to the rate control mechanism to form a packet pair. The receiver uses a median filter on the intervals between the two packets in a packet pair to estimate link capacity and sends this value back with the ACK packet.

The sender side window size W is updated as $W = (W * 7 + AS * (SYN + RTT))/8$ where AS is the receiver side packet arrival rate looped back in ACK packets by the UDT receiver. The initial size of the flow control window W is set to 2 during slow start phase and is updated to the number of acknowledged packets, when the sender receives an ACK. The slow start phase used in UDT ends when a packet loss occurs or the maximum window size is reached, after which the congestion control algorithm starts to work. Packet spacing interval is set to 0 during the slow start phase and to the packet arrival interval, after slow start ends.

UDT protocol satisfies max-min fairness on single bottleneck topologies. On multibottleneck topologies, any flow is guaranteed to obtain at least half of its fair share according to the max-min rule. Particularly, the fairness of UDT is independent of path RTT in contrast to the classical TCP behaviour.

UDT is TCP friendly, however a regular TCP flow can occupy more bandwidth than an UDT one does, except if the network bandwidth-delay-product Q is very large, so that (Reno) TCP fails to obtain its fair share of the bandwidth[27]. In this situation, UDT will occupy the bandwidth which TCP cannot utilise. When short web-like TCP flows coexist with a small number of concurrent UDT flows, the effect of UDT on TCP flows is very small [52].

4.6.3 DCCP: Datagram Congestion Control Protocol

The Datagram Congestion Control Protocol provides a standardised framework (ACK format, path MTU discovery, identification, negotiation mechanisms, connection set-up and tear-down, security) for implementation of other congestion control mechanisms. It does not stick to certain congestion control behaviour by itself, however additional DCCP documents for specific congestion control are available.

At the moment of this writing, there are two profile definitions available for DCCP: Internet draft draft-ietf-dccp-ccid2-02.txt for TCP-like congestion control and Internet draft draft-ietf-dccp-ccid3-02.txt for TFRC congestion control. TFRC is an equation based congestion control mechanism which is TCP-friendly and shows a smoother rate than TCP but with similar fairness properties, therefore able to coexist with TCP in the public Internet. More information on TFRC can be found at [56].

4.7 Conclusion

The TCP protocol despite its age is still the dominating source of traffic on the global Internet. This is a huge triumph of TCP and ist versatile architecture which proved over the last 30 years to cope well with the continuous development of internetworking technology. The popularity of TCP however cannot be only contributed to its wide presence in virtually any operating system. Moreover its basic congestion avoidance and detection algorithm is doing "the right thing" in most situations. Nevertheless the community discovered that with the growing popularity of less reliable and slower wireless type links classical TCP may (under certain circumstances) suffer from insufficient performance.

Therefore it is not an accident that there is an increased effort in the research of improved and more sophisticated TCP algorithms as well as specialised queueing solutions for TCP over WLAN links with the increasing prevalence of those problematic wireless technologies. The development of TCP goes mainly into three directions as follows.

First there are ongoing efforts in improving overall TCP performance (especially

[27]since its average CWND size may on average stay lower than the fair-share CWND size due to the slower and slower window expansion for large Q.

from the perspective of improving its effectiveness of congestion avoidance and its stability in equilibrium) and its hop-by-hop ressource utilisation regardless of the exact type of interconnecting network between the end points (Vegas-TCP, ECN-TCP). It has been recognised by the community that simplistic Reno-like congestion control may even end in a global congestive collapse of the Internet [43].

Second main research direction looks for more specialised TCP extensions, for example to improve the end-to-end performance of TCP on those high-error wireless links (like Westwood-TCP, Jersey-TCP [135] and Veno-TCP [47]), or to accelerate TCP in sense of the achievable effective throughput over very high speed networks like ten gigabit Ethernet (Scalable TCP or HyBla TCP) where classical Reno-TCP is known to perform rather poorly.

Third interest area of current research but with far less significance (and therefore out of the scope of this work) for this thesis is enhanced mobility support in TCP. While TCP has not been designed as a protocol supporting mobility of end hosts, there is more and more demand for such extensions with nowadays increasing dynamism of the mobility of Internet users (e.g. between workplace and home). Nonetheless WLANs like IEEE 802.11 are believed to require rather micro-mobility support which is rather a MAC layer issue (cell roaming).

Last but not least classical TCP can be seen as just one protocol from many being reliable but stream-oriented, with urgent notification and no multihoming, implementing a benign congestion avoidance algorithm not harmfull to the network along its transmission path under normal circumstances. Some of the TCP features however may not be required or may even be undesirable for certain types of applications (e.g. its byte-by-byte stream in case of a video media transfer, where frames must be delivered on time) so that there is an increased effort to separate the building blocks of TCP into independent entities and provide generic solutions e.g. for datagram congestion control and avoidance like the work done with DCCP and UDT.

The next chapter presents a short introcuction into the actual state of research in the area of wireless TCP performance and discusses the encountered problems.

Chapter 5

TCP in Wireless Environments

5.1 Introduction

An average wireless environment exhibits a number of undesirable properties, of which the designers of older TCP algorithms like Reno were not aware or not concerned about at the time TCP was developed:

- Wireless network bandwidth is usually scarce and far below the bandwidth of a LAN backplane network used to connect the wireless gateway to the company infrastructure or public Internet. Moreover, a wireless access gateway will likely serve just few clients in contrary to a core router, which will probably have to deal with hundreds, if not thousands of parallel TCP connections. It is not very clear, how standard TCP sources behave in such scenario.

- Wireless transmission is prone to channel errors and failures and may lead to a significant packet drop rate (compared to a wired-type link). Since TCP interprets packet losses as network congestion, the result is a substantially degraded TCP performance over wireless links. This issue has been addressed by numerous papers and the classical solution proposed in literature is to try to hide wireless packet losses from the TCP congestion avoidance algorithm. A few of those proposed solutions are described later in this section.

- The available wireless network throughput usually changes with the physical location of the mobile node, since the quality of the radio signal heavily depends on the terrain topology and distance from the base station. Therefore nodes in a WLAN cell may experience different network performance at a given time, resulting in unfair short- and long-term bandwidth sharing.

- In a 802.11-like network we must also consider that the communication channel is only half-duplex (in theory it would be possible to have full duplex radio units however the 802.11 standard adopted only half duplex radios for cost reasons), so that in contrast to a wired link, TCP data packets transmitted by the AP to wireless nodes may affect (delay, collide or even corrupt) ACK packets traversing the channel in the opposite direction.

- A negative impact of wireless MAC retransmissions on TCP performance has been postulated in the literature (however up to my knowledge never measured and reported to appear in a real setup). The performance degradation is expected to come from concurrent MAC- and TCP-level retransmissions influencing each other (since the TCP source is unaware of lower level retransmissions), if performed on an improper time scale[1].

- Finally, the impact of the inherent randomness present in the 802.11 MAC CSMA/CA access method on real-world TCP performance must be investigated in more detail.

5.2 TCP over Lossy Links

Since classical TCP congestion control assumes that packet loss results from congestion in the network path, it is expected to perform badly on a wireless-type link with high bit error rates [28]. Three different directions have been proposed in the literature to address the high error susceptibility of TCP in wireless environments and can be classified as `split connection` solutions, `link layer` solutions and sender or receiver side modifications to TCP. A comprehensive comparison of those solutions has been given in the evaluation work [97], and I want to give only a brief and exemplary discussion of the key features of those proposals.

5.2.1 I-TCP: Split Connection Approach

The `I-TCP` or `indirect TCP` protocol [11] was the very first proposal utilising the split-connection approach. The key idea of I-TCP is to logically split each TCP connection to/from a mobile host into two independent TCP connections (that is basically to proxy) at the base station: one TCP connection between the BS and the outside (Internet) stations, and a second TCP connection between the BS and mobile nodes, using standard TCP of whichever kind is available at the BS node. The assumption made by I-TCP approach is that rather the outside nodes but not wireless nodes are TCP senders most of the time and that therefore the splitted connection can shield the TCP sender side from wireless channel errors thus preventing wireless transmission errors from influencing sender's congestion window size and the overall throughput.

While the split connection approach have also an additional advantage in case of wireless micro-mobility[2], the choice of standard TCP for the wireless part of the splitted connection is not very effective. If due to link errors the wireless side of the splitted connection times out, it may also make the other side of the connection stall and let it enter slow start via the TCP timeout mechanism, therefore influencing connection's throughput exactly the same way as if the connection would have been made directly between the two end hosts.

I-TCP also imposes at least doubled total processing overhead compared to a single TCP connection, which must be performed by the base station. Since the two

[1] **e.g. the MAC queue could be flooded with retransmissions of a TCP segment while trying to retransmit the segment itself.**

[2] **that is mobility inside a wireless distribution system, so that the outside TCP endpoint can be shielded from potential change of wireless IP address by keeping the AP-endpoint and rerouting the connection between APs.**

split-connection endpoints at the base station are both independent TCP instances and just use the built-in TCP stack in the AP machine, data from the original connection may be packetised differently on the wireless side breaking the end-to-end principle[3]. Each TCP packet causes also a checksum computation and verification at the base station which may quickly reach its CPU processing limits even with a powerfull machine.

The end-to-end semantics of TCP is also broken on the reverse path (from the AP to outside world), since with two independent TCP connections ACKs may even reach the outside sender node before corresponding data segments are able to reach the mobile node at the wireless end of the connection. This effect is in some sense orthogonal to possible sender stalls in case of wireless errors, since it may permit the outside TCP to send faster than the available wireless bandwidth. The I-TCP solution is also related to reliable sockets (Rocks) introduced in the paper [140].

5.2.2 Snoop-TCP: a Link Layer Solution

The `Snoop` ([17], [16], [15]) protocol belongs to the second kind of solutions proposed to combat the high bit error rate problem in context of wireless TCP. The Snoop protocol introduces a TCP cache module called the `snoop agent` at the base station. The Snoop agent monitors the TCP stream of packets at the base station for and from each mobile destination and maintains an IP stack independent cache of past TCP segments, that have been sent to mobile nodes but have not been acknowledged yet. A wireless packet loss is detected by the duplicate ACK mechanism like in classical TCP or by a per-packet timeout. In case of packet loss or error the Snoop agent will retransmit the missing segment and, if necessary, suppress the duplicate acknowledgements, therefore effectively shielding the TCP sender from any wireless channel errors.

The local cache of the agent only needs to maintain a soft state and can be constructed in the middle of a TCP connection (e.g. after a wireless handoff[4]), but may on the other hand demand a lot of additional memory at the base station, if numerous mobile stations with more than just few TCP connections should be served. No significantly increased per packet processing overhead is imposed at the AP node, on the contrary to I-TCP, because the Snoop cache contains only ready-to-send TCP segments.

The performance of the Snoop approach for different TCP variants (Tahoe, Reno, Newreno, Vegas and SACK-TCP) have been evaluated in the original paper [128] and the Snoop approach found to perform well with TCP versions other than Vegas. On a Snoop improved wireless link Vegas seems to perform badly, if more than four packets are lost in a wireless error burst. TCP Vegas is also reported to exhibit the worst performance among these above 5 different TCP variants in the absence of the Snoop agent and in the presence of frequent wireless link errors. A concurrent approach to Snoop called TULIP using reliable MAC-level delivery instead of reliably

[3]some applications rely on TCP push to timely deliver interactive data like SSH, etc.

[4]a hard state transfer may be difficult since it may require kernel-level data to be transmitted between machines, which is not easy even for same kernel software and rather difficult if the architectures differ.

delivering TCP segments over an unreliable MAC layer has been proposed in the articles [105] [106]. The disadvantage of Snoop in direct comparison with TULIP is that it does not work for encapsulated TCP like TCP carried in a VPN tunnel due to unavailability of TCP header information at the access point.

5.2.3 SACK-TCP: Selective Acknowledgements

While solutions based on a split-connection or link-layer approach usually require only a modification of the base station or an intermediary node, TCP level solutions only work, if either the mobile nodes or corresponding TCP senders at the other side of the connection (or both) are modified. This is not very comfortable, since while it may be possible to force all the users in a wireless cell to install a special OS extension[5] on their machines before they are allowed to use wireless resources, it fails completely if all of the Internet machines and servers must be modified. However, several proposals have been made to include some modifications to TCP in its future implementations and one of them is the selective acknowledgement scheme (found in [72], [92], [42]). For example the Linux kernel already includes a working SACK implementation.

The cumulative acknowledgement scheme of standard TCP sender does not permit an effective loss recovery, if more than one packet is dropped from a window of data. The TCP sender can only repeat transmissions of what appears to be the last successfully received segment, until it is acknowledged and then it can proceed with the next missing sequence number. It takes at least one RTT to retransmit the missing segment in standard TCP once a loss has been detected, so that if more than one segment is lost from the actual CWND, it takes multiples of this time for classical TCP to recover from a bursty loss. It is even worse, since once the first missing segment has been retransmitted, standard TCP will enter fast recovery and reset its duplicate ACK counter, so that it must wait for another three duplicates to retransmit the next missing segment and this means another three RTTs of waiting.

On the contrary, in the original SACK RFC [92] the authors propose to include in each ACK packet information about up to three non-contiguous blocks of data so far received (including information about most recently received segments in the first block), therefore permitting the sender to retransmit just the missing segments in one turn. In addition to Reno, SACK-TCP has a new working variable called `the pipe` and an internal data structure called the `scoreboard`. The pipe is incremented when the sender sends a new or a retransmitted segment. It is decremented when the receiver receives a new data segment. The scoreboard stores ACK numbers from recent SACK blocks, allowing the sender to retransmit those packets first, which appear to be missing at the receiver with the highest probability.

A real-world analysis and evaluation of SACK-TCP has been performed in the paper [14], where authors evaluate TCP traces obtained from a real and busy WWW server and report that SACK-TCP does not improve upon Reno-TCP in majority

[5]and the development of such an extension may be pretty strenuous if the source code of the operating system is not available!

of situations for the reason of the insufficient congestion window size or severe congestion, preventing enough of duplicate ACKs to arrive. The same observation is made in the second SACK paper [13].

5.2.4 TCP Decoupling: TCP Connection Shielding

A novel approach to the high sensitivity problem of regular TCP to non-congestive losses has been proposed in the interesting paper [84] and called `TCP Decoupling`. The article discusses a method to decouple the transmission of TCP header information from TCP packets used to carry user data. Instead of attempting to distinguish congestive packet losses from those caused by errors on traversed wireless links, the approach uses tiny (about 40 bytes in size) header packets to implement TCP congestion control for a separate stream transporting larger data packets.

Since the per-packet packet error rate (and therefore the probability that a packet is lost or dropped later due to incorrect checksum) is directly related to the MAC-level packet size (as mentioned previously) and the size of the header packets is small, the authors propose to base the congestion control only on information carried by header packets but not on the actual data volume transported by the datagram stream. By introducing this modification to TCP and exploiting the different probabilities for big datagram and short header packet loss, authors report to achieve TCP goodput improvements of up to 350 percent over regular SACK-TCP and Reno-TCP for high-error wireless links.

Unfortunately, the authors fail to realise that a decoupled TCP as proposed in the paper introduces a very high overhead in presence of a 802.11 type wireless link, where the real cost of a very short packet (like a TCP control packet) is as huge as about 50 percent of a 1500 byte datagram and erroneously report an additional overhead of only about 1.7 to 3.4 percent.

The advantage of decoupled TCP over the two previous solutions is its preservation of the end-to-end nature of regular TCP. The main disadvantage of the decoupled TCP approach is that it requires support from each of the communicating hosts, since at the datagram level decoupled TCP is incompatible with the standard version. Therefore TCP decoupling is not a suitable approach for ad-hoc or hot-spot wireless networks like in rail stations or airports (where random hosts are supposed to access the wireless network) and probably will not be deployed in practice (at least not so soon).

5.3 Conclusion

An average wireless environment exhibits a number of undesirable properties, of which the designers of older TCP algorithms like Reno were not aware or not concerned about at the time TCP was developed. Classical TCP congestion control assumes that packet loss results from congestion in the network path. Therefore it is expected to perform badly on a wireless-type link with high bit error rates. A TCP connection is expected to underutilise the wireless link in presence of very frequent transmission errors. Similarly the end-to-end delay is expected to show high variability due to frequent TCP recovery through timeouts. It is rather improbable that future development of the WLAN technology will permit to quickly mitigate those specific wireless issues and permit to achieve acceptable performance with oldish-style TCP like TCP (New)Reno being still the prevalent TCP version nowaday.

Thus three different directions have been proposed in the literature to address the high susceptibility of TCP to transmission errors in wireless environments and can be classified as split connection solutions (I-TCP), link layer solutions (Snoop) and sender or receiver side modifications to TCP (SACK-TCP, TCP decoupling and Westwood-TCP described in the preceeding chapter). I think that the two first research directions are interesting but more of theoretical interest due their rather high complexity and overhead. However the two solutions (I-TCP [11] and Snoop [17]) discused above show that the problem of degraded TCP performance over high-error wirless links can be solved locally (up to some extent) by a specialised TCP queuing/cache module aware about the presence of the wireless hop. Nevertheless the third direction seems to be more promising because wireless links are just a special case of an unreliable link and the research should try to solve the TCP performance problem at the layer where it origins from: the TCP congestion avoidance and detection algorithm itself.

This chapter finishes the general state of the art section of this work and prepares the ground for the analysis of practical (that is observed in experimentation as well as in simulation) 802.11 performance of out-of-the-box 802.11 WLAN devices and their (probably unwanted) interaction with a higher-level protocol like TCP, which is the subject of the following two chapters.

Part II

Analysis of Performance Problems in WLANs

Chapter 6

Evaluation of 802.11 MAC Performance

6.1 Chapter's goal

The purpose of this chapter is not to provide an exhaustive measurements of the 802.11 MAC performance, since such results have been already published in the articles [61] [62], but rather to look at the practical implementation of 802.11 in actual wireless products and to show that an "average-only" method (that is looking only at the long-term throughput of a single WLAN card) is not sufficient to quantify a networking device incorporating such a complex technology like IEEE 802.11. Moreover this chapter provides a brief description of the experimental testbed and the developed benchmark software used in the following chapters, although later chapters use also NS2 simulations to obtain the results.

6.2 Hardware Setup Description

While the previous chapters were devoted to the presentation of the state of the art in the wireless networks and QoS, I want to proceed with providing some experimental results obtained with consumer 802.11b hardware. Since the IEEE standard is not always unambiguous and some features are just optional, one may not assume that all, theoretically 802.11-conforming wireless devices will show equal performance in practice.

There are basically three different types of currently available 802.11 wireless equipment: consumer cards which may support 802.11a/b as well as the g version of the standard, but not able to operate[1] in the wireless master mode[2], premium consumer cards with master mode available (where cards with the PRISM2/3 chipset like Compaq WL100, Linksys belong to) and built-in hardware of stand-alone AP equipment[3] too numerous to be listed here. WLAN cards with master mode usually also offer a WLAN monitoring mode, where any received WLAN frames are forwarded to the device driver and can be used to analyse the WLAN at the frame level.

[1]or with no driver software currently available which provides AP functionality.
[2]which permits building of a software access point.
[3]embedded devices.

The experimental data presented here was collected using the hard- and software configuration described below. A managed WLAN cell was set up using a Linux machine (P2, 350 MHz) running the kernel version 2.4.27[4] acting as a software AP. The wireless card used to manage the cell was a PRISM2 based Compaq WL100 card together with the HostAP version 0.2.1 of the driver software. The mobile nodes used for experimentation were a number of Dell Latitude D600 (PentiumM at 1400 MHz) machines with built-in Intel Centrino WLAN cards (802.11b), however if necessary, replacement cards were used via the integrated PCMCIA slot.

6.3 Benchmark Software Description

In order to obtain trustworthy experimental data (available tools like TTCP or Netperf either do not work well, e.g. in UDP mode, or their output is not comprehensive enough to study wireless network dynamics), two wireless benchmark programs have been developed: CBR and HTBench, both intended for use with the Linux operating system and which I want to describe briefly.

6.3.1 CBR: Constant Bit Rate Source for Linux

The simpler of the two programs is a constant bit rate (CBR) UDP source called just cbr, which can operate in receive or send mode. The packet source is constrained by two token bucket filters (TBF) responsible of peak and average traffic rate shaping respectively. In send mode an output rate (together with an optional peak rate) is provided at the command line and maintained at the UDP socket level[5].

Cbr's send mode offers an automatic bit rate sweep from a specified minimum to a maximum rate and also permits an end-to-end round trip time (RTT) estimation[6]. The RTT estimation works by marking (in the packet body, which carries otherwise no useful data) some of the regular UDP packets as PING packets (if not otherwise changed at the command line, cbr will use roughly 10% of the packets for RTT measurement) and which are returned to the source by the cbr listener. Each PING packet carries a time-stamp generated from the sender's system clock at the time, at which the packet has been sent to the kernel socket. Usually such measurements are carried out under a circular route setup, where the source and destination node have an alternate (usually Ethernet) path to each other, so that the wireless medium is not affected by the returning PING packets (which means that if the delay over the Ethernet line may be neglected, the measurement of the RTT can be considered as one-way end-to-end). If in RTT estimation mode, cbr also allows to randomise the spacing between PING packets[7] and this may be important, if a bursty but periodic

[4]since the Linux kernel is developing rather fast and also wireless drivers tend to change their versions rather quickly, it is not possible and even not recommended to always use the latest one because I want to preserve the experimentation environment.

[5]it is important to note that the two tools provide an end-to-end benchmark, that is, including delays and jitter introduced by the OS, so that some of the results presented here may slightly depend on the software and hardware configuration used, however permitting a more realistic estimation of network bandwidth and delay.

[6]for RTT measurements the cbr listener must be started with -R option.

[7]but still using the same percentage of packets.

source is present on the wireless[8]. Another useful feature of cbr is the ability to save all of the RTT samples into a single file therefore permitting to analyse the distribution (e.g. histogram) of network delay.

In cbr's receive mode, a socket is bound to any desired UDP port and a measurement of incoming UDP data bandwidth is performed[9]. The base time interval for bandwidth estimation[10] is freely configurable and usually set to few seconds. The measured effective bit rate at the socket level can be corrected for the effective rate at the IP level taking the header (IP+UDP) lengths into account. If the RTT estimation mode is switched on, the cbr listener will return its averaged receive rate inside the echoed PING packets, therefore permitting to easily collect all of the experimental data (RTT and bandwidth) at the sender machine[11].

The cbr packet generator cooperates with IPv4 as well as with IPv6 version of the IP protocol. The command line of the cbr tool has been shown below and is self-explaining[12]:

```
lap001:~/QoS # ./cbr

CBR v1.20
USAGE:  ./cbr OPTIONS
options are:

        -6 IPv6 mode
        -h <host>
        -p <dstport>
        -s <srcport>
        -r <rate>
        -M <max. rate (sweep)>
        -T <sweep step time>
        -n <sweep steps>
        -a <or sweep rate add>
        -l <packetlen>
        -b <burstlen>
        -H correct for IP+UDP header length
        -u <update int sec.>
        -R RTT measure
        -U randomise ping packets
        -N <ratio of RTT packets>
        -P <ping queue depth, none for auto>
        -F save all RTT values into <filename>
        -S do not print head line
OR:
        -L listen mode
        -s <port>
```

[8]mitigation of any possible synchronisation between sources.

[9]that is the effective data bandwidth at the destination station is measured.

[10]since bandwidth measurement means an averageing over some time interval, while in theory the bandwidth is the first derivative of transferred volume over the time which is complicated to calculate in a packet based system.

[11]this is important since the sender side rate for UDP has no meaning in a WLAN because of possible high packet drop rates.

[12]a pair of <> braces means a parameter value.

6.3.2 HTBench: a Concurrent HTTP Benchmark

In contrary to the cbr UDP source, which is a custom-protocol application, the HTBench tool was designed to permit easy studies of application level performance and dynamics of TCP-based connections and intended to work with any server software using a simplified version of the HTTP protocol [129] (therefore it is a client-only software). The internal architecture of the HTBench application consists of an asynchronous multi-socket engine driven by a contextual state machine in a single thread of execution and it works as follows.

A benchmark configuration file is parsed and defines a number of URLs for resource locations (`request classes` in HTBench nomenclature). A resource location definition specifies the physical as well as the virtual location of the resource[13] and the method used to access the resource. It additionally defines the local IP used to make the connection[14]. The method may be either a HTTP GET to request a download of a particular resource to the local machine, or HTTP PUT to upload a local data file (actually the file is emulated in memory and consists of random bytes[15]) to the server. In addition the number of concurrent requests for a particular resource must be given in the configuration file[16].

Once the configuration file has been read the benchmark is performed[17] until the configured number of requests has been made. During the benchmark run, each request is done in an asynchronous manner (connection establishment, HTTP handshake and data upload or download) and a per-request result file is written to the local storage.

The result file permits easy study of TCP dynamics and contains a time-versus-volume (that is, the up to the time t acknowledged TCP data volume $V(t)$) table gathered during the lifetime of each TCP connection. A time-volume diagram is the preferred method for studying TCP dynamics, since the bandwidth of a TCP transfer is a time dependent observable which also depends on the selected base time interval. However, connection's bandwidth can easily be obtained by calculating the first order derivative of $V(t)$. The time measurement begins with the successful TCP connection establishment therefore containing the time for HTTP handshake and application latency at the server side, but the time for TCP's three way handshake is not counted[18].

Since HTBench was designed to ease benchmarking of server applications under realistic conditions, it is for example possible to simulate the behaviour of an average LAN user where some amount of background traffic (like longer FTP/HTTP downloads or media streaming) must compete with interactive sessions made of short

[13]that is the physical IP of the server together with HTTP virtual host and network port

[14]for multihomed machines.

[15]in my experimentation I used a NULL-CGI at the server side which just discards its input.

[16]the concurrency may be of course 1 that is one request at a time and there must be at least one request class in the configuration file.

[17]actually HTBench permits broadcast synchronisation, that means it may wait until an UDP broadcast packet to a specified port appears on the interface therefore permitting easy synchronisation between multiple wireless machines. A tool for generation of arbitrary IP packets including UDP broadcast can be found on the web [75]

[18]in order to provide reliable measurements, a number of coding tricks has been used, like memory pre-faulting and asynchronous I/O.

requests (like WWW page views) by just providing an appropriate configuration file. An integrated permutation scheduler inside HTBench's state machine minimises any effect of connection synchronisation[19].

A sample HTBench configuration for a wireless machine with two physical wireless cards to upload and concurrently download a file from a WWW server machine has been shown below and is self-explaining:

```
set max_traces 2
set sync_bcast 1
10.10.10.2 10.10.10.1 8001 pitcairn /file_10mb.html 1 8192 get_cent_%d.dat
10.10.10.3 10.10.10.1 8001 pitcairn /cgi-bin/post.cgi 1 8192 put_compaq_%d.dat
                                    PUT 10485760
```

6.4 Experimental Results

In this section a short evaluation of the wireless cards which were available during my work is presented. In the first experiment the conformance of various 802.11b cards with the IEEE standard has been investigated. In order to gather experimental data, the mentioned software AP machine has been equipped with a Compaq WL100 card. Analogically one of the available Dell portables has been equipped with a PCMCIA Cisco Aironet 350, Compaq WL110 and a Intel Centrino IW2100b (built in) wireless cards respectively.

6.4.1 802.11 DCF Conformance Test

A simple 802.11 conformance test was carried out as follows. On the AP machine the tcpdump [73] utility was running in the background to capture all wireless packets while the card inserted into the mobile node had to send one million of UDP packets 1024 bytes each with the maximum possible speed (that means that the 802.11 MAC layer of the WLAN card was always backlogged). The mobile node was the sole source of wireless traffic during the test. The output of the tcpdump command containing the packet trace together with arrival time stamps[20] was then saved to a file and a statistical analysis was performed as follows.

For each consecutive pair of packet arrival time stamps the positive time difference (delta) is calculated[21] and frequency counting of the delta values with a bin resolution of $\Delta = 1\mu s$ is carried out. Finally, the resulting arrival delta frequency table is normalised to represent the probability of an arrival delta in each of the equally sized $(N * \Delta, (1 + N) * \Delta), N = 0...MAX$ time intervals.

The result of the statistical counting can be seen in 6.1 and can be interpreted as follows. At about $T = 1.6\mu s$ a number of sharp peaks can be seen in the diagram for each of the cards used in the test. The peaks correspond to the CSMA/CA

[19]that means that in each service cycle TCP connections that have their state changed (e.g. new data arrives, socket buffer space available, etc.) are served in random order.

[20]the arrival time stamp is the Linux kernel system time stamp at which the packet was passed by the card driver to upper layers.

[21]this is done automatically by tcpdump if run with -ttt command line argument

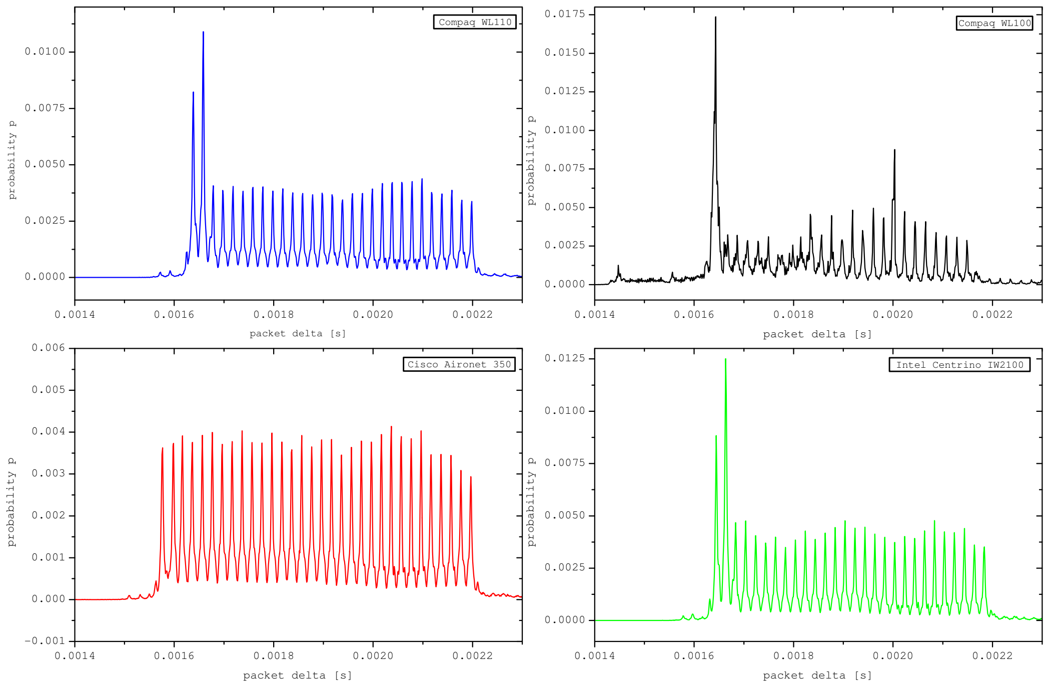

Figure 6.1: Experimental 802.11 conformance test #1: inter-packet spaces for a maximum speed UDP source.

access method previously described and can be related to consecutive DCF time slots during the backoff period after each packet transmission. Since the MAC was always backlogged during the measurement, the system built of the sender, wireless MAC and receiver node works like a pipeline, with its output frequency (that is in this case the packet rate) only limited by the duration of the longest operation (with duration T) which is the wireless transmission time itself[22].

For a stream of fixed size packets however, the only difference in the wireless transmission time can arise from the variable number of backoff slots a station had to wait before each packet transmission[23]. By exploiting this observation one can sample the contention window of the wireless card and take a look at its PRNG[24] used to select backoff slots[25]. The noisy period before the first visible peak in each of the diagrams represents the duration of the physical packet transmission time over the air. The noise in the experimental data comes from randomness in the involved subsystems (OS kernel, memory bus, IRQ latency, etc.).

[22]the application sending/receiving packets, the IP stack of the machines, the wireless hardware and the channel are all independent subsystems in the sense of packet queueing.

[23]remember from previous sections that a node must follow the backoff procedure after each successful transmission.

[24]pseudo-random number generator.

[25]since there is only one card sending, the residual backoff time equals the original backoff time selected by the wireless MAC.

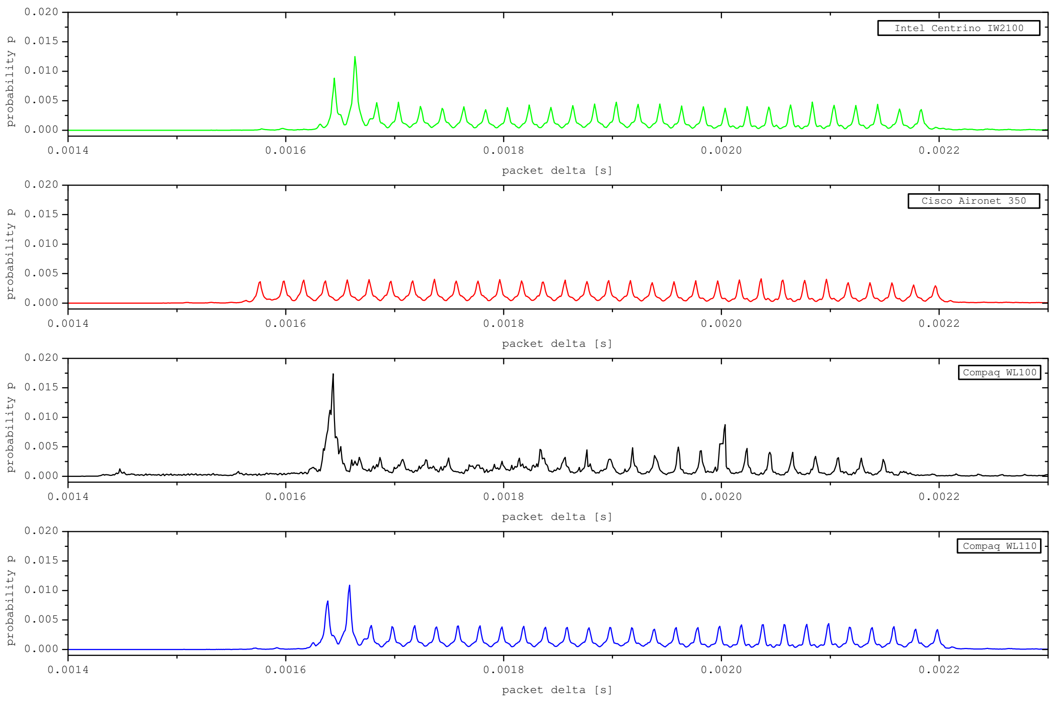

Figure 6.2: The same data as in figure 6.1 shown in a single column.

To simplify the comparison of the 802.11 cards the same data as in the figure 6.1 has been reorganised and presented in the figure 6.2. It is clearly visible that three of the four cards do not fully comply to 802.11: the WL110, WL100 as well as Centrino cards have troubles with their PRNG algorithm used to select backoff slots and either some of the mandatory 32 slots[26] (since there are no wireless collisions with just one card sending) are missing, or (or as well) the distribution of backoff slots is not uniform[27]. Only the Cisco card seems to follow 802.11 specification, however if the PRNG deviates from "perfect" behaviour on a smaller time scale, the deviation won't be detected by this kind of test[28].

To further test the cards' 802.11 conformance, a second experiment has been carried out as follows. For an UDP CBR source constrained at 32 kbyte/s and sending 1500 byte sized packets, same statistical delta counting analysis was carried out. The result for the Cisco card is shown (the other cards delivered an equal result) in the figure 6.3. It is clearly visible that now a constrained source can benefit from the low-delay property of the DCF access method, if it has rarely a packet to send. The reason for three and not just one peak in the plot is the behaviour of the CBR source itself, since it uses a token bucket filter to constrain its output rate, which

[26]or the firmware setting for the `CWMIN` differs from the default value of 802.11b.

[27]of relevance is only the integral probability over a $20\mu s$ sized time slot which is not uniform for the three cards too.

[28]e.g. if the "PRNG" outputs a constant sequence like 0,1,2,3, ..., 31.

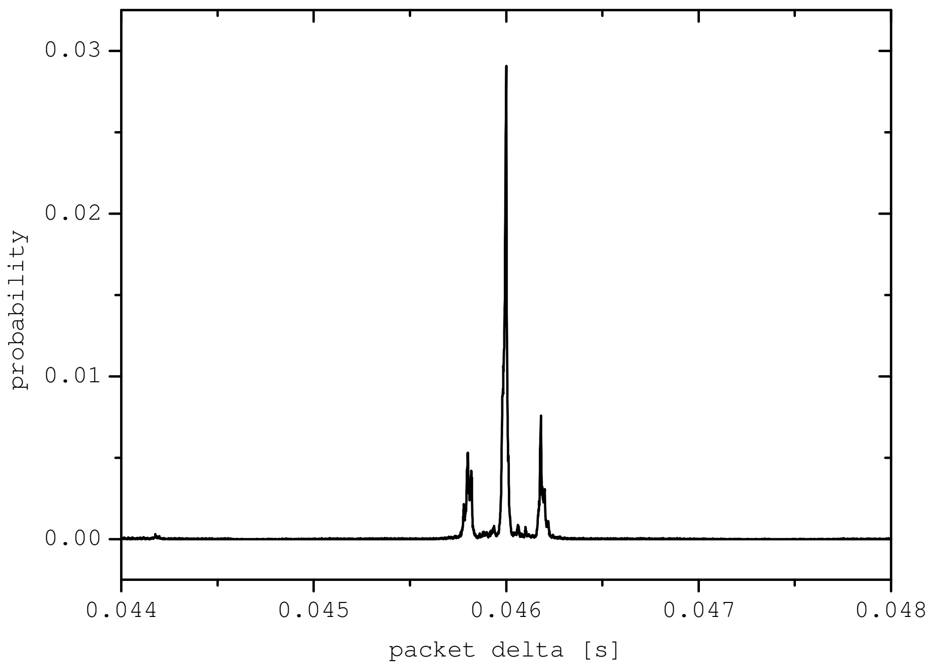

Figure 6.3: IEEE 802.11 conformance test #2: inter-packet spacing for a 32kb/s CBR UDP source.

works by means of the system clock having a limited resolution. In order to maintain a given sending rate, the source determines a sleep time interval necessary to get enough tokens permitting sending of the next outstanding packet. If woken up by the OS kernel, the time which has passed while the CBR process slept will be used to calculate the amount of new tokens available in the TBF, but for the reason of the limited clock resolution this may be a little bit more than necessary to send the next packet. Since the additional tokens will stay in the TBF and will be used later, an alternating sequence of packet spacings around the optimal sending time appears.

6.4.2 CSMA/CA Equal Access Probability Test

Another basic 802.11 conformance test has been conducted to investigate the 802.11 CSMA/CA access method with respect to channel access fairness and carried out as follows. One mobile node equipped with a Centrino card was set up to generate a high-throughput best-effort (BE) like UDP flow composed of 768 byte sized packets. Second machine equipped with a Compaq WL110 card was used to generate a low bandwidth but constant bit rate 64 byte packet stream (denoted with EF) and the cbr source was programmed to increase the output bandwidth from 8 kb/s (that is 64kbit/s) to 128 kb/s by 8 kb/s per minute. The effective throughput as well as the arriving packet rate was measured at the access point node for each of the two traffic sources.

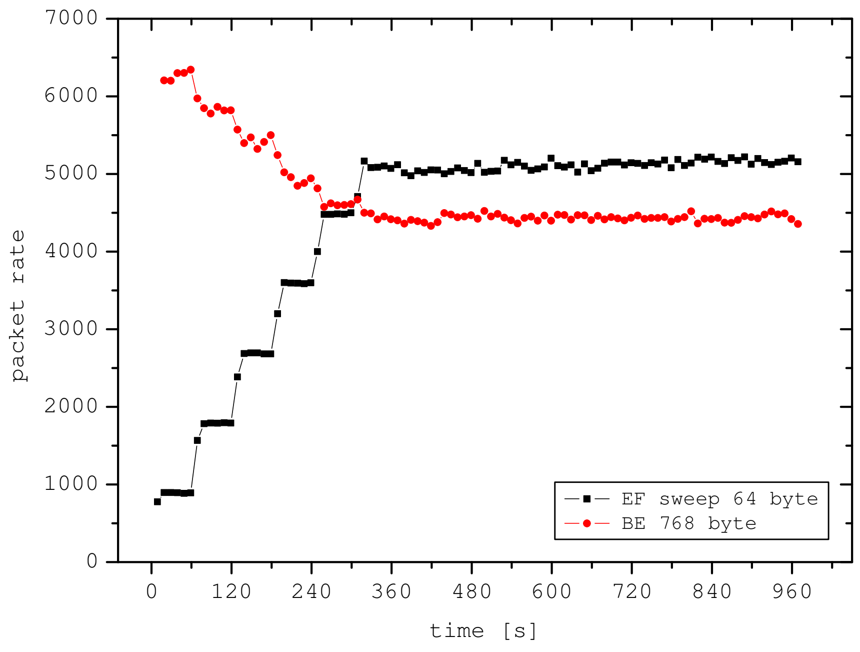

Figure 6.4: 802.11 CSMA equal channel access probability test.

The result is shown in figure 6.4 (only the packet rate is shown, throughput of each of the sources can be obtained by multiplying the packet rate with the respective packet sizes) and can be interpreted as follows. While at the beginning of the measurement the BE source can fully utilise the WLAN channel capacity, its effective packet rate drops quickly with increasing packet rate of the second source.

The packet rate rate of the BE source does not drop as quickly as the rate of the second one grows, since with two active 802.11 MACs less time is spent in the contention, therefore raising the maximum achievable channel packet rate[29]. While the BE source alone is not able to send more than around 6500 packets per second the total WLAN packet rate with two active stations raises to about 9500 packets per second, that is by about 50 percent! As will be shown later by simulation, for just very few nodes[30] a 802.11 WLAN exhibit anomal behaviour, which can be quantified by the introduction of an effective window size.

As one could already expect from the previous conformance test and the DCF histogram, the packet rates do not equalise as they should in theory, if the cards fully complied to the 802.11 WLAN specification. The discrepancy is of the order of about 20 percent and seems to be significant. The lower rate of the Centrino card can be attributed to broken PRNG implementation on the card. Repetitions of this

[29]however collisions impose a counter-effect on the achievable packet rate but for two nodes collision probability in 802.11b is as little as about 3 percent, see later.

[30]which may be a relevant scenario in practice since 802.11 WLANs are intended for hot-spot or home use, where few nodes (<10) are expected in the cell.

kind of experiment with other card combinations which I had at hand at the time of this writing convinced me that 802.11 products are really far from being perfectly interoperable and also far from delivering equal and fair performance.

6.4.3 Throughput Tests

Another basic WLAN experiment has been conducted to measure average UDP and TCP application level throughput achievable with the three cards[31] and summarised in the following table 6.1: The result is not really surprising and it could already

Table 6.1: Average WLAN throughput

	Compaq WL110	**Centrino IW2100**	**Cisco A350**
UDP 1472 bytes	720 kb/s (unstable)	723 kb/s	729.1 kb/s
UDP 64 bytes	99.6 kb/s	97.7 kb/s	97.1 kb/s
TCP download 10MB	589.1 kb/s	603.8 kb/s	615.1 kb/s

be expected from the outcome of the conformance tests: the Cisco card delivers highest throughput since it better utilises the regular 802.11 contention window and is therefore able to send some of its packets with less delay than the other WLAN cards. The WL110 card exhibited slightly unstable behaviour during my test runs and the UDP throughput dropped sometimes below the average value[32].

It is also clearly visible from the above table that a 802.11 like LAN is very ineffective for small packet sizes and this observation can be attributed to the high per-packet overhead imposed by 802.11, which consists of the slow-rate PLCP header, inter frame spaces and the inevitable backoff time. This is a very important "feature" of IEEE 802.11, since a common scenario where majority of WLANs get deployed includes Internet gateway functionality and therefore the traffic in the wireless cell is dominated by protocols commonly used on the Internet.

It has even been studied in literature ([33]), that the dominating source of Internet traffic is TCP, however this fact is also well known from personal experience of people using Internet each day. Therefore it is reasonable and legal to make the prediction that the average traffic in a WLAN cell is most probably composed of long data packets carrying TCP payload in one direction as well as of short packets for corresponding TCP ACKs.

6.5 Conclusion

This chapter provided a very basic evaluation of existing 802.11 devices. The experimental finding is that although the 802.11 is an IEEE standard, conformant devices do not show equal performance in practice. Depending on the particular vendor of

[31] only one station was sending.

[32] it is worth noting that in early tests which I conducted obtaining stable WLAN measurements was more a matter of luck than of know-how for the reason of unstable and buggy driver and OS software.

the WLAN card the achievable throughput and packet rates may vary. This observation has been related to the implementation of the random number generator on WLAN cards by using a method to fingerprint the contention window and obtain the probability of slot selection in the contention window. Thus even if two different cards may be able to achieve nearly equal average throughput for a simple UDP packet source being the only one source of traffic in the wireless cell, the packet rates don't need to equalise, if the two cards are put in direct competition with each other.

Therefore a measurement of classical performance parameters like average throughput and delay is not sufficient to characterise the quality of a 802.11 WLAN card. Moreover the conformance with the standard must be tested by applying the fingerprinting procedure described in this chapter. Nevertheless the statistical fingerprinting captures only the average behaviour of the WLAN device being unable to quantify short time-scale deviations from a perfectly random behaviour. This issue is left for further research (e.g. χ^2 testing [35]).

Another conclusion of this chapter is that actually 802.11 transmission is very inefficient for small packets. This could be expected from the description of the DCF access method presented in chapter 3 and can be attributed to the huge overhead wasted in the channel access and the acknowledgement of the physical frame. This is an important observation in the context of assumed WLAN utilisation for seamless Internet connectivity, where the majority of the traffic originates from TCP thus being composed of data frames being rather long (for larger data transfers) but also of very short ACK frames in the opposite direction, both in a rather unpredictable mix. Therefore the goal of the next chapter is to analyse and observe the behaviour of a 802.11 cell in common but simple access scenarios which may result in such a packet mix. While 802.11 has been designed as a fair protocol at the long-term packet level rate (but see chapter 8 for explanations about its short-term fairness), it may not be automatically expected that a complex protocol like TCP, where the application data is carried only by a part of the MAC level packets (namely only those with a data payload) will perform fairly from end to end over a 802.11 half-duplex link.

Chapter 7

Problem of TCP Asymmetry and Fairness in WLAN

7.1 Wireless Access Scenarios and TCP Behaviour

While previous research in the area of TCP over wireless concentrated on the high error rate problem prevalent in wireless environments, which is surely an important point, there are more issues specific to TCP and the behaviour of its congestion control algorithm related to wireless especially like imposed by IEEE 802.11 networks, which have not been or have been insufficiently recognised by researchers.

Although the problem of MAC-level short-term fairness between wireless nodes has been recently addressed by [60], the achieved packet-level fairness improvements are still not able to resolve the problem of the upload-download asymmetry in TCP. Since an optimal 802.11-like MAC provides fairness only at a per-station packet rate level, concurrent TCP upload and download streams from N wireless stations ($N-1$ mobiles and an access point) sharing an IEEE 802.11 wireless channel may suffer from substantial throughput unfairness.

This postulated phenomenon can be understood as follows. While any of the $N-1$ wireless mobile stations can (on the average) benefit from the about $\frac{1}{N}$-th fair share for channel access at the packet level, the aggregated flow of all download streams directed to some (or all) of those wireless nodes can for the same fairness reason utilise only the $\frac{1}{N}$-th packet rate share of the access point node. The packet level rate fair share of each node must be certainly distributed among the upload and download flows of the station, since in case of classical TCP any flow will induce a non-negligible stream of ACK packets in the reverse direction.

In order to present a vivid example of such `mixed upload-download scenario`, assume that there are N active wireless nodes, for example M nodes with one active TCP upload and $(N-M)$ nodes with one active download stream at the same time. In this situation one can expect the effective bandwidth for each of the TCP upload flows to be close to the $\frac{1}{M+1}$-th part of the usable wireless channel capacity[1] B,

[1] **with "effective bandwidth share" I mean the fractional part of the end-to-end bandwidth obtained by a flow with respect to the total end-to-end bandwidth obtained by all flows crossing the AP. This is certainly less than the gross WLAN capacity, e.g. less than 11Mbit for a 802.11b WLAN as well**

whereas the effective bandwidth for a TCP download flow to be close to the roughly $(1/M+1)^2$ fraction of B.

This asymmetric bandwidth distribution comes from the fact, that not all of the participating wireless nodes can be backlogged at the MAC level at the same time in this scenario, even if the TCP/IP layer (socket) is fully backlogged with application data. While a station performing a TCP download waits for a data packet and then confirms the packet with a TCP ACK, a station performing a TCP upload may have a full window of data (if permitted by the receiver of course!) waiting for transmission and may continuously contend for the wireless channel. That means in other words, that downloading stations work in a polling-like mode, while uploading stations are more probably permanently backlogged and always participate in the channel contention in each DCF cycle.

One may argue that this can not be fully true, since uploading stations can only be backlogged if the destination TCP returns ACK packets, which must also pass through the bottleneck access point node, therefore the situation should be self-regulating. However, due to cumulative character of TCP ACKs, uploading stations need to get just few of them (that is acknowledging a sufficient portion of the effective sending window) in order to be able to advance their sliding window and feed the wireless MAC with new data packets.

If the composition of upload and download flows gets even more complex, with arbitrary number of flows per station in any of the two directions, the exact behaviour of TCP and its interaction with 802.11 MAC can be hardly predicted. However, two scenarios seem to be relevant in practice [12]:

- a hot-spot WLAN (in an airport or hotel), with mostly anonymous users, which enter and leave the cell occasionally and perform routine tasks, like WWW browsing, Email access or maybe have some interactive sessions (company login via SSH). Such a scenario is characterised by a highly varying amount of parallel network users, while TCP sessions being rather short and sporadic and more likely of the download type.

- infrastructure access WLAN, like a company-wide LAN replacement network. Such scenario can exhibit higher amount of upload-type TCP flows which can origin from VoIP, video conferencing or other interactive communication software, while requiring certain minimal amount of bandwidth in the download direction for software updates, file server access or just web browsing (also video conferencing and VoIP is usually full duplex).

Whichever the exact demand may be, it is obvious that assuring some bandwidth guarantees for the upload but also for the download direction is a central QoS criterion which WLAN designers are concerned about. Wrong QoS provisioning and performance of a WLAN installation may lower the acceptance of end-users while making it even impossible to deploy certain types of applications.

known from 802.11 literature.

7.1.1 Simple Download Case: Verification in NS

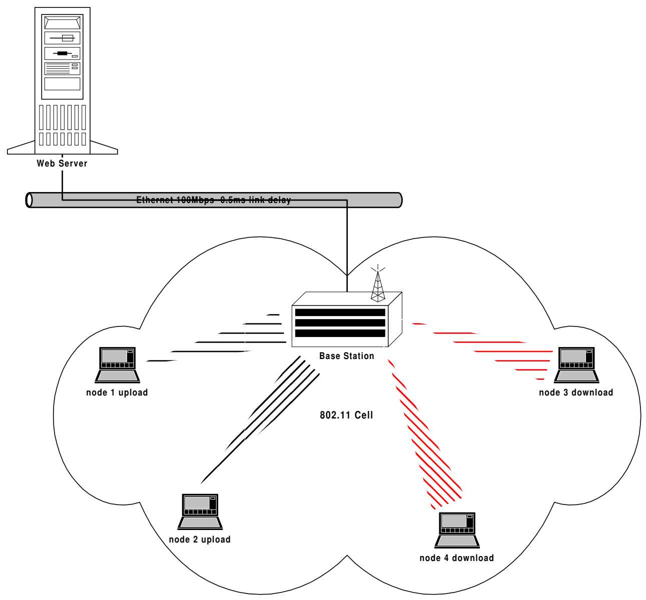

Figure 7.1: Wireless network topology used in NS2 simulations

To verify and confirm the preliminary considerations about the performance of the classical TCP in a wireless environment, a number of simulations was ran in the NS2 network simulator version 2.28. A wireless access/company network was modeled by the topology shown in the picture 7.1 and was used in all following simulations. The integrated 802.11 module of the NS2 simulator was used for the wireless MAC, however, it was reconfigured to be closer[2] to the IEEE 802.11b network standard. The relevant MAC level parameters were set according to the following table 7.1.

Table 7.1: NS2 802.11 module MAC settings used in simulation.

802.11 variable	**Value**
$RTSThreshold\_$	3000
$ShortRetryLimit\_$	7
$LongRetryLimit\_$	4
$CWMin\_$	32
$CWMax\_$	1024
$slotTime\_$	20 μs
$ifs\_$	20 μs
$sifs\_$	10 μs
$pifs\_$	15 μs
$difs\_$	20 μs
$PreambleLength\_$	144 bytes
$PLCPHeaderLength\_$	48 bytes
$PLCPDataRate\_$	1.0e6 bits/s

[2]**Since the default configuration of NS2' 802.11 module resembles a 2Mbps FHSS PHY.**

Because of the chosen setting for the RTSThreshold value, virtual carrier sense via NAV/RTS was disabled during the simulation. The simulations did not study channel errors and node mobility, since those issues have well known and well written literature [28], [136], [63], [115]. In all simulations NS2 PriQueue (priority queue) class with preference to routing protocols was chosen for the queueing discipline at mobile nodes and the maximum permitted queue length was set to 100 packets. Common in all simulations is a queue size of 10 packets per active node (that is, for 6 nodes it is 60 packets in length and so on) in the access point (otherwise the comparison between different numbers of stations won't be fair).

The TCP algorithm used in the simulation is NS2 implementation of Newreno, which is close to what is available in the Linux kernel (for comparison with practical experimentation later). The initial congestion window size of all TCP sources was set to 4 segments and the maximum window value was bounded by 32 segments (that is similar to values which Linux version 2.4 is using[3]). The initial three way TCP handshake was also turned on in all simulations, in order to obtain more realistic results. To present the behaviour of TCP sources the highest so far acknowledged segment number over time is shown in following pictures, which if multiplied by the TCP segment size (set to 1500 bytes) gives the transferred TCP volume as function of time.

In the first simulation (scenario 1) the behaviour of 6 concurrent download flows

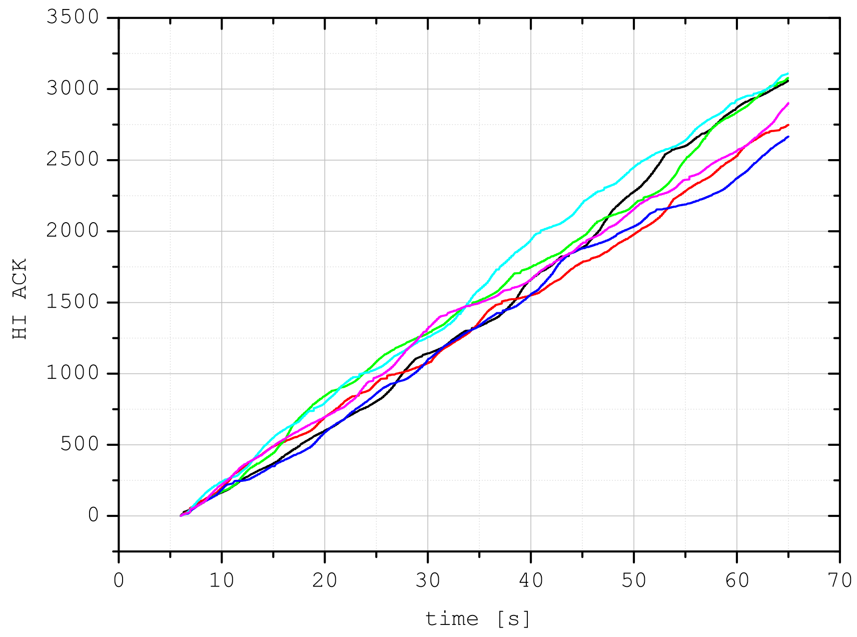

Figure 7.2: Scenario 1: simultaneous TCP downloads, 6 NewReno wireless stations, transferred volume shown.

[3]Other OSes cover the range permitted by RFC from 1 to 4 MTU segments [103].

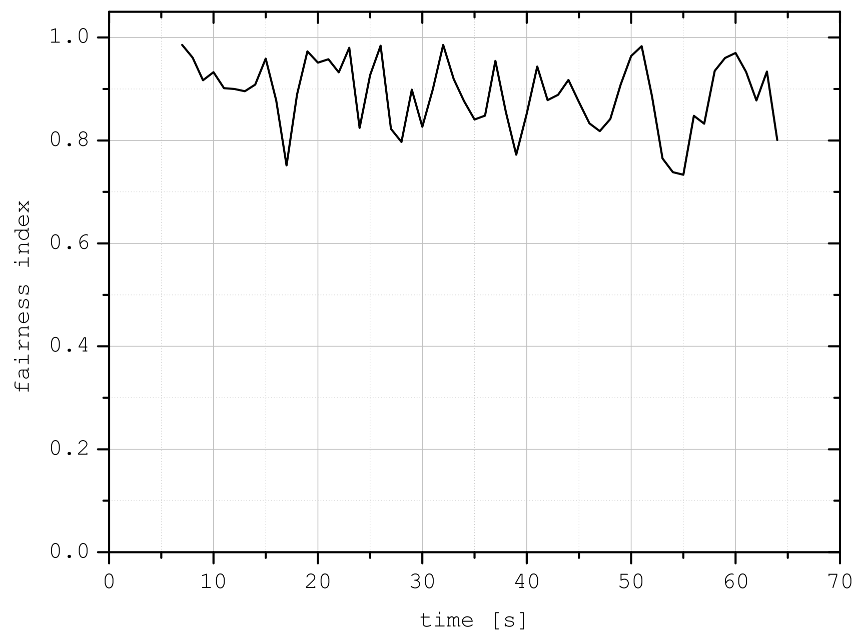

Figure 7.3: Scenario 1: Jain's fairness index computed with a resolution of $\delta = 1s$.

for a drop tail queue at the access point was studied to obtain a reference simulation. In the volume graph shown in the figure 7.2 it can be observed, that a significant variation in the TCP download bandwidth over time results in this simple scenario. The strength of the variation can be quantified by applying the Jain's fairness index J [74] to the TCP download volumes of N nodes:

$$J(t) = \frac{(\sum_{i=1}^{N} \Delta_i)^2}{N \sum_{i=1}^{N} \Delta_i^2} \tag{7.1}$$

where Δ_i represents the TCP volume increase during a time interval T (which was set to 1s for calculating the fairness plots). The index J is very sensitive to even small variations in the download volume of different stations. The fairness graph shown in the figure 7.3 confirms, that simultaneous Reno downloads suffer from substantial unfairness, even without the presence of any other upload flow. This is maybe surprising at the first glance, but can be understood as follows. An ensemble TCP-Reno sources can operate roughly in two distinct regimes:

- a steady state region where all of the stations can always send their maximum window of data on the network. A real TCP source[4] is always constrained by the amount of buffer space available at the receiver (flow control) and once the window size reaches this maximum possible size, the bandwidth will remain constant[5] in the absence of any packet losses. Under normal operating condi-

[4]**Attention in NS.**

[5]**to combat this problem window scaling option has been introduced into TCP.**

tions (real network, random background traffic, etc.) this regime is relatively rare. For example Linux has a default socket buffer size of roughly 200 kbytes and this amount of data (approximately, since the data is encapsulated into socket buffers at the kernel layer which add overhead) can be kept in transit for each TCP flow. This is a huge bandwidth-delay product reachable only for roughly gigabit links!

- a chaotic or sometimes also cyclic regime, where the congestion window experiences continuous changes due to congestion avoidance, fast recovery or TCP timeouts.

In the second regime roughly three different types of (Reno-) TCP behaviour can be observed as explained in the analysis [111] and hold for TCP flows with similar RTT values crossing a single congested bottleneck. Depending on the number of concurrent connections N and the total packet buffer capacity W of the bottleneck link (bandwidth-delay product) and its associated queue (that is the total number of packets the link and the queue can hold) it can be differentiated into:

- `large pipe case`: defined by the relation $W > 3 \cdot N$ and dominated by congestion window synchronisation between the TCP sessions. The explanation of the synchronisation behaviour is the simultaneous loss of data segments, once the bottleneck queue overflows. The sufficiently large size of the network pipe permits all sessions to have (on average) roughly 3 data segments (therefore the above factor of 3) queued at the bottleneck, thus recovering in synchrony from the congestion event by three duplicate ACKs[6]. The buffer occupancy at the bottleneck node oscillates between the maximum and roughly a half. Since the congestion windows of all connections are synchronised, all of the sessions are going to receive their fair share of the bottleneck bandwidth.
- `intermediate pipe case`: occurring for $N < W < 3 \cdot N$ and characterised by local synchronisation of congestion windows, that is synchronisation between subsets of active TCP sessions. Since not all of the TCP flows can store at least 3 packets in the bottleneck queue at the same time, then if the queue overflows, some of the TCP sessions will recover from the loss by fast retransmit while the others will wait for the retransmission timer to expire. Due to network randomness (at least in the wireless case), the set of connections using fast retransmit and the set using timeout mechanism will change over time, thus no global synchronisation can occur. On a much larger time scale than the scale at which the sets of the two types of sessions change, the transferred volume may still be relatively equal for each connection[7], therefore maintaining long-term fairness but no short-term fairness between the flows.
- `small pipe case`: observed for $W < \cdot N$ and also characterised by local (but less prevalent) window synchronisation but also by a number of probably shut-off (stalled) connections. In this regime new TCP connections (that is connections opened after the regime has been reached at the congested link) are potentially going to receive very little or even zero bandwidth.

[6]This holds for TCP using 3 DUP-ACKs to recognise congestion, which is a configurable parameter e.g. in Linux.

[7]If they persist sufficiently long.

Applied to the wireless scenario simulated above the operating point which is going to happen in practice is the small to the medium pipe case. A relatively small queue is desirable and recommended at the access point, since a 802.11 WLAN is rather slow but relatively low-delay. The exact link delay of 802.11 WLAN is not just a single value but depends, due to the DCF access method, for example on the number of contending stations and may exhibit a high jitter value for even very few contending nodes. Under low to moderate load the average packet delay is of the order of one millisecond for a 802.11b WLAN operating at 11Mbps, that is the bandwidth-delay product is roughly 11 kbit.

Thus an AP-side (as well as mobile side) queue of just few packets[8] seems enough to keep the link busy, however due to the potentially higher jitter of a 802.11 WLAN a bigger queue is recommended. On the other hand increasing the buffer capacity at the AP too much may introduce additional delay in presence of Reno-TCP. In Linux for example, the default queue length for network interfaces (txqueuelen) is 1000 packets but this far too big for a 802.11 WLAN, since a single aggressive Reno connection crossing the AP can fill such a long queue and induce queueing delay of several hundreds milliseconds resulting in a very bad performance for other interactive sessions (like SSH and Telnet).

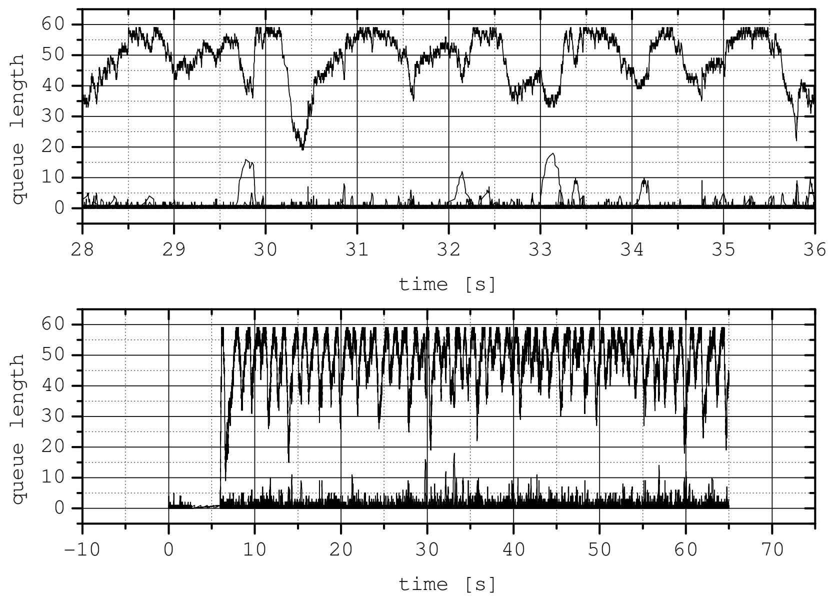

Figure 7.4: Wireless scenario 1: queue occupancy for AP and mobiles. Upper plot shows a zoom into the lower plot.

The above considerations are confirmed by the queue length (figure 7.4) and `CWND` (figure 7.5) plots from the first simulation. In the resulting operating point the AP queue overflows periodically, but not very regularly, suggesting the average pipe case. Once the queue is full, every outside TCP source which sends a packet now,

[8]**that is at least those 11 kbit.**

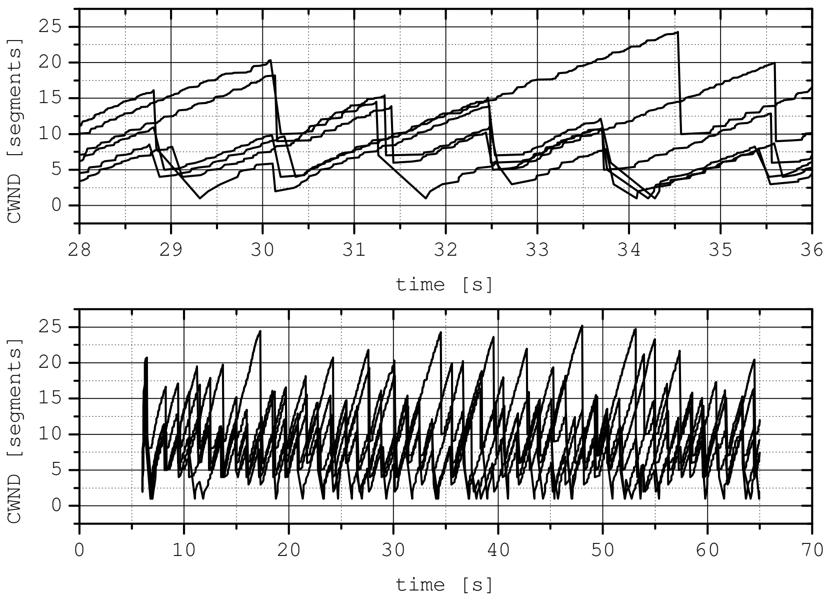

Figure 7.5: Wireless scenario 1: congestion window size evolution with local synchronisation (upper plot zoomed).

will have that packet dropped and will probably detect congestion by duplicate ACKs. The synchronisation effect is not as strong as for a regular deterministic link, because the 802.11 random access method breaks the synchronisation a little bit. As observed in the article [111] adding random TCP processing delay (that is randomising packet send outs at the source) may break window synchronisation, if the amount of randomness is sufficiently high. The article [111] states for the large pipe case that a variation of roughly 10% of the total path RTT is sufficient to break global synchronisation and turn it into local synchronisation. For a 802.11 WLAN operating at 11Mbps a relatively small queue of only 10 packets can already contribute about 90% of the total delay, thus the randomness of 802.11 DCF is not sufficient to substantially influence the TCP behaviour, at least not in the case that more than just one mobile node is active.

In this context it is also interesting to mention the impact of RED queueing on a congested wireless link. While RED is believed to break/prevent global synchronisation at core Internet links, it may have a negative impact on a wireless link, since both DCF and RED add randomness to the process of packet forwarding in a non-coordinated way (and usually the sum of two random variates is less random than the variates themselves). While the normal effect of DCF is just a random packet delay, RED drops packets randomly, however under certain conditions (like in a highly loaded WLAN with a lot of stations) the jitter of 802.11 may be perceived as packet loss by TCP sources, if the retransmission timer expires. The result of combining of these two effective packet drop event sources is not predictable.

I did not achieve any substantial fairness improvements by replacing the drop tail queue at the AP by a RED queue in NS simulations for reasonable RED parameter settings. The nominal effect of RED queueing can be understood as a reduction of the effective queue size at the AP and moving the operating point of the link even more towards the medium to small pipe case. RED can in fact solve only the AP queue overflow problem but not the fairness problem. RED is not easy to configure and requires at least three parameters to be set to "the right" values: L_{min}, $L_{max}\ and p$ for the minimum queue length (at which RED starts dropping packets), the maximum queue length (where all packets are being dropped) and the initial drop probability (the drop probability for the minimum queue length) respectively. Unfortunately in a wireless LAN with highly variable number of users suitable values for those parameters are not easy to find. RED is better suitable for a core link where short-term traffic parameters are not too volatile.

Another NS2 simulation also confirms, that the described unfairness effect vanishes, if the queue size at the access point is made bigger and bigger, however a large queue size at the bottleneck is undesirable, since it impacts the performance of interactive sessions and may have an impact on new connections arriving at the bottleneck (since their TCP doesn't have any idea about the network state yet, new connections will show degraded performance at the very beginning). The optimal operating point of a network link can be defined to be the point, where there is always a next packet to send in the queue if the link layer becomes idle, but the congestion doesn't build up too much.

Unfortunately, this requirement is not compatible with (New)Reno's (as well as with Westwood's) behaviour which depends on periodic queue overflows to detect actual network capacity. While on a core network bottleneck with a large queue size (but a very fast link) the additional congestion induced by one more Reno connection may be just few (or even fractions of, on average) packets, on a relatively low speed wireless link with just few active sessions at once it may significantly increase the average queue size. Therefore in such a situation Reno-like TCP is going to measure mostly its own congestion (self-congestion) but not the permanent congestion induced by the aggregate packet flow from other sessions at the bottleneck and its congestion avoidance algorithm fails.

A delay based congestion estimation like in Vegas TCP seems to be more promising at the first glance in the few-nodes scenario, since a growth in the measured RTT value is a good indication of bottleneck queue build up. In a real-world application there is a practical obstacle: all of the TCP senders which could communicate with mobile nodes in the wireless cell must be upgraded to TCP Vegas and this is not an option, if the wireless cell should be more than just a laboratory setup. Moreover, even if a significant number of Internet and company machines one day incorporate TCP Vegas into their TCP/IP stacks, a single Reno connection will still overflow wireless bottlenecks and aggressively outperform and degrade the performance of other active Vegas flows. Therefore my belief is that even in a download-only (and few node) wireless network scenario a specialised wireless active bottleneck queue management is necessary.

Second caveat of Vegas TCP in the few-nodes wireless access scenario is related to the Vegas estimation of the expected bandwidth: since Vegas measures and main-

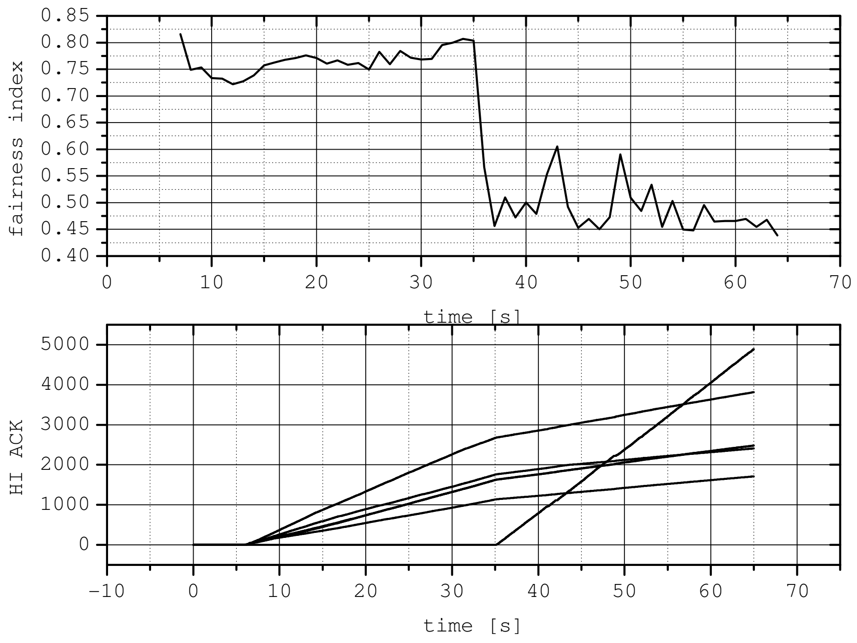

Figure 7.6: Scenario 2: Vegas(2,4) and NewReno transfer volume and fairness index, TCP download

tains a BaseRTT value, that is the minimum round trip time ever observed during the lifetime of the connection, some Vegas flows may not be able to get a realistic estimate of this value, because they enter the wireless channel after some steady congestion has already built up at the AP. Thus some of the competing Vegas flows may have a much lower estimate of the expected bandwidth than others, resulting in long-term unfairness (in contrary to Reno which suffers more from shorter-term unfairness). Although Vegas is in principle configurable via its α, β, γ values, it is difficult to find a suitable setting for those parameters which would improve fairness in a given wireless scenario ([131]).

It is important to note that the literature reports contradictory results for Reno and Vegas performance and fairness behaviour and this can be contributed to the variety of scenarios and network topologies studied by researchers. For example [26] reports that fairness of Vegas is at least at the level of Reno, however the authors compute the fairness index only for the duration of the complete connection, that is they only consider long-term fairness.

A sample simulation for (New)Reno and Vegas concurrence situation (scenario 2) has been shown in the figure 7.6 and confirms the findings about Vegas-versus-Reno behaviour cited in the established TCP literature. Due to its aggressiveness Newreno-TCP outperforms simultaneous Vegas downloads by a huge factor, in the case of 6 competing wireless nodes even by a factor of about 10 times the volume received by Vegas flows. Moreover, what is also visible from the plot 7.6 is the mentioned inability of Vegas to establish fairness at the very beginning of the simulation without the additional Reno flow. The catastrophic fairness behaviour can be contributed to Vegas' static link estimation algorithm which performs badly in the case of just few concurring sessions (absence of active bandwidth probing in the stability region).

Another observation can be also done in the simple download scenario: taking a look at the figure 7.4 that shows queue length variations over the time, an effect related to 802.11 access method can be recognised. The queue length curves show the queue length for all participating nodes including the AP node over the total simulation time. AP's queue is obviously (on the average) the longest one, while mobile queues stay rather short during the simulation run. In this simple simulation scenario the mobile queues store only ACK packets generated in response to data packets leaving the AP queue. Due to the low-delay property of 802.11 DCF if station's MAC is not backlogged, ACKs are expected to pass through the wireless channel with no delay. However as the graph shows, sometimes ACK queue stalls occur, that is the ACK queue of the mobile is growing up, while at the same time the AP queue is decreasing (since a TCP source usually sends packets in bursts), indicating that data packets are being received by the mobile but its MAC can not send the next packet for a while even if backlogged. This can be contributed to the 802.11 short-term unfairness, where a station is not able to send packets after it has collided and expanded its contention window (backoff selection interval) too much. For this reason the mentioned ACK compression phenomenon[9] is relevant in such WLAN utilisation scenario.

In reality the situation is even more complex, since a real TCP is not so comfortable with sending ACKs like a simulated one. In simulation an ACK can be generated immediately, without any delay from layers which exist in a real OS. In contrast, on a Linux PC equipped with 802.11 hardware a packet received by the 802.11 MAC must first pass the card hardware, that is the MAC payload must be checksummed and transferred to the host memory. Depending on the type of wireless hardware, this may involve a generation of an interrupt introducing additional latency. From my experimentation I know that a 1GHz P3 Linux machine can serve at most about 200.000 interrupts[10] per second. A hardware interrupt is even more costly, since it may involve a PCI bus transaction in the host.

The travel of the TCP data segment is however not finished yet - it must pass all the TCP/IP stack layers that require some time for processing the packet, before it can reach the corresponding application socket and trigger the ACK generation[11]. The ACK packet must be first allocated at the kernel layer, that is a memory allocator is called, packet memory cleared and a TCP header written to memory. And so on and so forth, finally producing a ready-to-send ACK packet passed to the MAC with a significant delay.

While that processing time elapsed, the wireless MAC was idle and a substantial amount of time may have passed, of the order of several 802.11 DCF time slots. That means that another station (in a download-only scenario, most probably the

[9]Relevant to TCP Westwood.

[10]this was measured by executing a tight loop triggering a software NOP interrupt, being therefore the absolute lower limit for real IRQ latency.

[11]on Linux from kernel 2.4 on, it is even more complex, since network processing is synchronous or asynchronous, a load-delay trade-off is used, keywords SoftIRQ and keventd mechanisms.

AP station again), which had a packet waiting may already have started the transmission. Thus some of the TCP ACK packets will have to enter the DCF channel contention procedure, but sometimes the mobile may be lucky and have the ACK packet ready before the backoff counter of other backlogged wireless nodes reaches zero. This implies that TCP ACKs are going to experience higher jitter than in the case of a full-duplex channel transmission and this is also one more reason, why Westwood TCP does not perform much better than Reno in a few-node WLAN download-only scenario.

7.1.2 Mixed Upload-Download Scenario

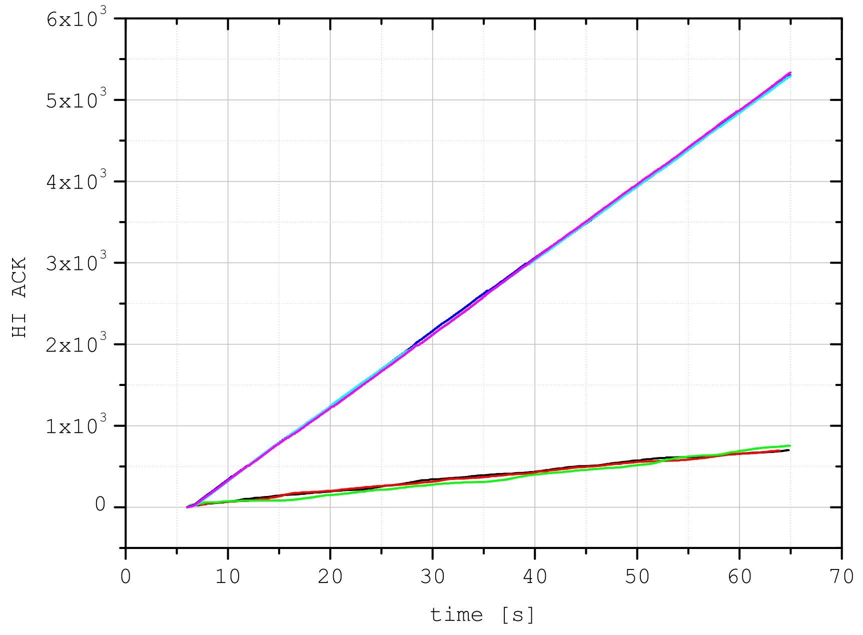

Figure 7.7: Simulation scenario 3: three stations with one TCP download (low bandwidth) and three with one upload (high bandwidth) under DropTail at the AP.

In order to study the postulated download-upload asymmetry effect in a 802.11 WLAN, another NS2 simulation has been prepared. Three of the sessions have now the packet flow direction changed, that is there are 3 nodes with one TCP download and 3 mobiles with one upload session (scenario 3). Other simulation parameters (NS 802.11 MAC) and the network topology have been reused.

As depicted in the volume plot in the figure 7.7 as well as in the fairness index plot shown in the figure 7.8 a significant unfairness effect in the long-term end-to-end throughput between the download and upload directions can be observed. This is obviously a different problem than the Reno-TCP inherent unfairness in the chaotic/cyclic regime previously described (which in fact holds only for one transfer direction) and can not be explained by the aggressiveness of the Reno congestion

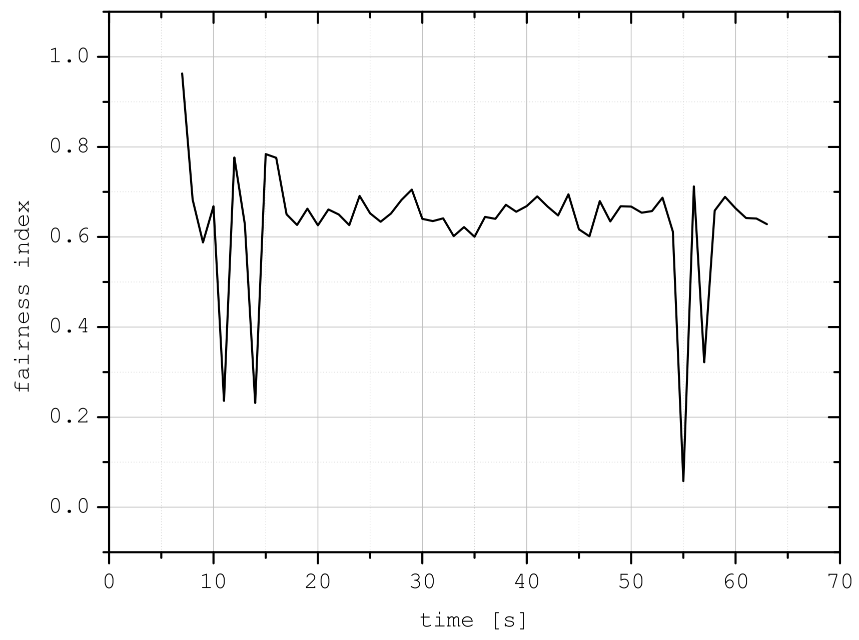

Figure 7.8: Composite Jain's fairness index for scenario 3 (3 upload + 3 download stations, resolution 1s).

control algorithm only.

The inherent unfairness of Reno is visible as a slight throughput variation between the TCP sessions for the same data flow direction, but on the other hand a huge bandwidth difference between up- and download sessions can be observed and contributed to the packet rate fair share property of the 802.11 MAC layer and the different susceptibility of the two session types to AP queue overflows as already expected from the section preliminaries.

While a TCP source operating in the download direction must pass each of its data packets by the shared queue at the access point, an upload-type source is fine with only a part of its ACK packets passing the shared queue due to the accumulative character of TCP acknowledgements. Thus an upload source is definitely less sensitive to the available buffer space at the access point. Therefore a TCP upload session can profit from the equal access probability at the 802.11 MAC layer, even if a significant amount of its ACK packets is lost during queue overflow periods at the access point. Any sufficiently high ACK packet (with high I mean an ACK packet acknowledging a big amount of outstanding data segments) can permit the uploading session to advance its sliding window and send a huge amount of data onto the network. On the other hand, a dropped packet in the download direction means wasted effort and the packet must be retransmitted by the sender.

The conclusion is that downloading nodes are penalised by packet drops at the AP, but uploading nodes can even profit from queue drops up to some extent, because their data is acknowledged by less ACK packets than in absence of bottleneck drops, leaving more usable network bandwidth for data packets. This is one more argument for an active queue management in the access point node.

7.2 Proposed Solutions for Fairness in WLANs

While in the preceding part of this chapter I've provided general considerations about WLAN-related performance issues with respect to the expected utilisation scenarios, I want to give now a short overview of other fair queueing solutions for WLANs proposed in the literature. A general approach at fair queueing in wireless networks has been presented in the two fundamental articles [114], [99] and provides a framework for adaptation of any (wireline) fair queueing algorithm to a high error rate WLAN-like network.

The first solution for WLAN fair queueing discussed here proposes a specialised 802.11 MAC level queueing algorithm, while the second solution proposes an IP-level algorithm well suitable for providing weighted allocations in the presence of real-time constraints.

7.2.1 DFS: Distributed Fair Scheduling

The DFS algorithm has been presented in the paper [127] and aims at solving the short-term fairness problems in a 802.11 type wireless LAN. The core idea of DFS is to use the `minimum slot property` of the 802.11 DCF backoff phase to implement a fair queueing algorithm in a completely distributed manner. The starting point of DFS is the centralised SCFQ (self clocking fair queueing) algorithm [48] applied to the wireless domain.

For implementing DFS the 802.11 backoff algorithm is modified as follows: instead of using a constant probability function to select a more or less random[12] slot after each packet transmission, the backoff slot is selected according to the node's next packet virtual finish time v_{end}. Therefore among all the packets contending for the wireless channel the one which would have been served under a centralised SCFQ will win the next contention cycle due to the `minimum slot property` of DCF. This is accomplished by setting node's backoff interval each time a packet with length L is passed to the MAC to:

$$b_i = \lfloor \rho \cdot S \cdot \frac{L}{w} \rfloor$$

where w stands for the wireless node's weight, which should be set to an equal value on all nodes in order to obtain fair bandwidth allocation; S being some scaling factor to span a finite number of regular 802.11 time slots; and ρ being a random variate with mean 1 to reduce the impact of collisions (set to uniform distribution over the $[0.9, 1.1]$ interval in the original DFS algorithm), if equally sized packets must be sent by two or even more nodes at once. As explained in the paper any global virtual clock dependence can be eliminated from the above equation.

[12]**Remember the weak PRNGs of current WiFi cards.**

A collision resolution procedure is also proposed in DFS: it uses an exponentially growing collision window and an uniform distribution used to select a random backoff slot after each collision event (an approach used also in the genuine IEEE 802.11). However, in order to better approximate a centralised SCFQ, the initial contention window is chosen to be small in order to yield backoff values less than any regular backoff value that can be obtained from the equation above by non-colliding stations. Thus the colliding stations can retry the sending of their packets before any other node seize the channel and so the collision resolution is effectively shielded from the regular channel contention. The authors remark however, that this kind of collision handling may lead to short-term unfairness in analogy to IEEE 802.11[13].

A very specific problem exists in DFS which in my opinion makes DFS not very suitable for practical deployment: the average idle wait time before each packet transmission. As the authors of DFS admit, their scheme works well, only if a sufficiently huge scaling factor is used in the DFS equation, that is enough time slots are available during the contention period for mapping of packet finish times to time slots. If the contention window counted in slots is too small, the scheduling becomes to coarse and too many collisions will occur, therefore ending in increased unfairness. Similarly, if the weights of nodes are chosen to be different but one of them is much smaller than the other weights, the DFS backoff scheme will automatically result in large average packet wait for that node, because the average packet wait is inversely proportional to the weight of the node and this is a very undesirable property. Desirable behaviour is to provide low delay as long as the packet load of the node is less or equal to its fair share of the channel with respect to the weight.

A non-linear mapping scheme is proposed for DFS to reduce the average packet wait time. Its goal is to map the backoff values obtained from the above equation to an effectively smaller number of wait slots, thus "compressing" the upper range of backoff values onto less time slots. An exponential and a square-root mapping is proposed in the paper, but each found to introduce potential bandwidth unfairness.

Under the DFS channel access scheme a trade-off between achievable throughput and short-term fairness is reported in the article. This will become logical after reading the considerations from my analysis of the classical 802.11 CSMA method in the next chapter. In reality DFS trades packet wait time for fairness. A similar result can be obtained with the standard 802.11 DCF, if the minimum contention window value is selected to be sufficiently big. That will keep the collision rate low and suppress the execution of the exponential backoff algorithm, which will be shown in the next chapter (by correlation analysis) to be the real source of short-term unfairness in genuine IEEE 802.11. Finally, DFS is missing a good adaptation scheme in order to deal with a dynamically changing number of wireless nodes (stations joining and leaving the BSS).

[13]**this will be explained by correlation analysis in the next chapter, which describes the importance of sending probabilities with respect to fairness!**

7.2.2 ELF: Effort Limited Fair Queueing

In the paper [41] a novel notion of fairness for high error rate wireless links called `effort limited fairness` is proposed. The key idea of ELF focuses on the insight that in a wireless environment one must distinguish between the "effort", that is channel transmission time spent in the attempt to deliver packets from a flow or station, and the "outcome", that is the actual useful throughput achieved by that flow or station. The notion is particularly interesting in the scenario, where some amount of high-priority flows carrying audio or video data and requiring absolute bandwidth reservation must coexist with background best-effort flows especially in the presence of location-dependent errors (that is packet error rates experienced by wireless nodes differing from station to station).

While in a wireline environment the effort usually equals the outcome, they can be substantially different in wireless environments. For example, if a wireless flow was assigned a fair share of 10 percent of the channel capacity but it experiences a packet loss rate of 50 percent, it will achieve an outcome of only 5% of the channel capacity. But if that flow is of critical importance for some application (e.g. VoIP) it may not make any sense to give it only the half of required bandwidth, which may render the application completely unusable, but it may make more sense to increase the effort spent on delivering those packets to achieve the required throughput.

An ELF scheduler strives to achieve the outcome requested by network users rather than the effort and this is accomplished by introducing limits on the effort spent on each flow together with a `power factor` setting. The power factor is a control knob which can be used to easily implement a variety of fairness and efficiency policies.

While a simple priority scheduler which serves high priority flows first could be used in such a mixed traffic scenario, the authors of the ELF algorithm argue that priority scheduling is too unfair to best effort flows and too simplistic in certain situations. For example assume, that the channel error rate increase so much, that even spending all the available channel time on delivering high-priority packets cannot satisfy the bandwidth requirements of the high priority flows, but would completely starve all best-effort flows. And since no flow would be satisfied, it would be an unjustifiable use (a waste) of channel capacity. In such situation a more intelligent distribution of the sending effort must be provided.

In contrast to a simple priority scheduler, an effort-limited fair server ensures that a reserved flow experiencing any transmission errors will fail to meet its reservation, even if repairing those errors would require only a small amount of unfairness to some other flow experiencing an error-free link.

The core component of the ELF server is a WFQ-like scheduler[14] controlled dynamically by an admission control module, which takes the MAC level outcome into account and works by modifying the WFQ per-class weights on the fly. WFQ weights w_i are adjusted according to a maximum `power factor` P_i that must be provided for each of the flows. The power factor defines the maximum effective weight that

[14]However for simplicity reasons the ELF paper evaluates only a WRR-based[37],[77] implementation of the ELF algorithm.

can be assigned to a particular flow and used to modify w_i:

$$w_i^{eff} = min(\frac{w_i}{1 - E_i}, P_i \cdot w_i)$$

where E_i stands for flow's i MAC level packet error rate[15] (a value between 0 and 1). On an error-free channel the ELF server behaves like the underlying WFQ scheduler. With increasing error rate ELF increases the effective weight, in order to match the outcome requirement of a particular flow and to cancel the effect of its reduced success rate $1 - E_i$. If all flows experience exactly the same link error rate, it can be shown that ELF also reduces to the underlying WFQ scheduler with original weights w_i running on an E-degraded link. On the other hand, if all flows are in their "effort region", that is have reached their maximum effective weight, the ELF scheduler is effort fair with respect to the power-adjusted weights.

By setting ELF power factors appropriately, the degree to which the fidelity of any flow will be maintained in presence of packet losses can be controlled. The power factor can be understood as the degree of importance of a particular flow. By choosing an absolute limit for power factors, one may control the scheduling efficiency on a lossy link, that is for example by choosing all of the power factors below 200% the minimum scheduling efficiency can be kept above 50%.

7.3 Conclusion

Table 7.2: TCP Algorithms Summary

	Newreno	**Vegas**	**Westwood**
Slow start	till loss	adaptive	till loss
congestion detection	loss	delay	loss
window control	rigid AIMD	proportional	adaptive AIMD
fairness short-term	bad	vary	medium
fairness long-term	satisfactory	vary	satisfactory

This chapter provided an introduction into the long-term fairness problem of TCP as expected to arise in modern IEEE 802.11 WLANs. There is a clear difference in possible utilisation scenarios of a wire-line ethernet and a wireless LAN which may lead to rather unexpected behaviour of well known protocols like TCP. This includes degraded performance (huge delay, unnecessary packet drops and timeouts) due to excessive queue occupancy at wireless bottlenecks, shut-off connections and underperformance of new (that is opened over an already saturated WLAN cell) connections.

Despite a huge amount of work done in research of improved TCP it must be concluded that regular TCP (meaning Reno and Newreno) as well as next-generation TCP (meaning Vegas and Westwood) does not always behave fair especially on short

[15]**Which must be available at the MAC level, however authors do not evaluate their scheme and with respect to real network equipment like IEEE 802.11., which do not provide such rates yet.**

time scales. While the long-term fairness of Reno-like congestion control seems satisfactory, Vegas-TCP may operate in an unfair regime for a life time of a connection.

The behaviour of classical TCP in wireless environments is influenced by the fact that wireless usually means "few-users" and "scare bandwidth". It is a known phenomenon (Aristoteles) in the nature that a large ensemble behaves differently than a system of just one or very few members of the ensemble - and while in a large Internet the impact of a single TCP connection on a core router can usually be neglected, it must be taken into consideration in such wireless environments.

Due to the unusual characteristics of wireless links especially those based on 802.11 (huge per-packet overhead, relatively frequent collisions, randomness of the MAC and last but not least MAC level fragmentation, reassembly and retransmissions - usually features of upper layers!) it must be concluded that classical transmission protocols behave at least differently (if not badly) compared to transfers over wireline links only. Moreover a drastically asymmetric performance of concurrent TCP upload and download transfers can be observed for genuine 802.11 WLANs (`upload-download asymmetry`).

The established literature provides fair queueing solutions at the packet level (that is either at the wireless link layer directly or at an intermediate bottleneck but only with respect to packet rates or bandwidth from flows or aggregates of flows) but rarely considers the outcome in the sense of the resulting end-to-end experience of the Internet user.

The goal of the following contribution chapters is therefore threefold. First, analyse the core of the fairness problem of collision avoidance schemes and try to deliver a more suitable quantification for fairness different from existing solutions (that is e.g. Jain's fairness index) but also to provide a generalisation of this class of algorithms. Second, the goal is to propose a solution for enhancing the observed TCP performance in such out-of-the-box 802.11 wireless environments, a solution which requires only minimal changes to the network infrastructure, optimally avoiding a change of the wireless clients. Third, the aim is to further investigate a possible change to TCP's congestion avoidance which may improve the short-term fairness.

Part III

Contributions

Chapter 8

Statistical Analysis of 802.11 and its Improvement

This chapter is aimed at providing new insights into the 802.11 DCF access method and its building blocks by investigating the general structure of collision avoidance schemes. This part of the work is also an attempt to answer the question about the structure of an optimal collision avoidance algorithm in the context of a wireless-type network. Recently, two interesting proposals to improve the basic 802.11 access scheme have been presented, namely `Idle Sense` which is a modified 802.11 DCF with congestion control at the MAC level [60] which is congestion control applied to the contention window size, and TCF [87], which is a TDM-based[1] distributed access scheme aimed at completely removing wireless collisions in the steady state for a constant number of continuously contending nodes.

8.1 Hash-based CSMA/CA

The 802.11b wireless network standard and the DCF access method have been broadly studied in the context of performance and its fairness in the recent WLAN literature [59], [21]. However, most of the studies focused on the existing protocol architecture and just tried to optimise its parameters (like the contention window parameters CW_{min} and CW_{max} or to fine tune the backoff algorithm (e.g. `slow decrease` [100]), etc.) in order to attain improved performance (lower the amount of collisions) or to modify IEEE 802.11 to better support real-time traffic. While all of the building blocks of IEEE 802.11 DCF (uniform probability for selecting contention slots, exponential and residual backoff) can be taken as given and optimised for any particular utilisation scenario, an attempt to understand the general structure of a CSMA/CA type protocol must be made and is the goal of this chapter.

Therefore if everything so far known and taken as granted about the 802.11 CA scheme is forgotten for a while, a principal question arises: "what is the best method to avoid collisions between N contending wireless stations on a shared channel?". Further assumption of the following consideration is that all wireless stations can hear each other and that their radio units used for transmitting wireless data are

[1]**Time Division Multiplex**

only half-duplex like those in 802.11 devices, that is collisions can be resolved only after the completion of packet transmission (by the absence of the corresponding ACK frame) and that simple carrier sense (no busy tone, etc.) is used. If someone would observe the access method of 802.11 from the outside lacking any information of its internal structure, he would make the observation that the radio channel access is a sequence of (time slotted) wait cycles which at the end yield either a packet collision or a packet transmission event.

8.1.1 Simplified H-table approach

An analogous situation where a similar collision problem arises is the hashing of key values into a discrete table of a given size. For this reason a radio channel access algorithm can be understood as episodes of key insertions into a hash table of some size. Each time some of the N wireless stations wish to send a packet, one may imagine that there is a virtual hash table with a number of virtual slots, where each of the backlogged stations "puts itself" by selecting a backoff value b^i_j and can be imagined as hashing of some station's key value into a slot of the virtual hash table. This can be formalised as:

$$b^i_j = H^i(i, j, k^i_j, C), i = 1 \dots N, j = 1 \dots \infty \tag{8.1}$$

where i stands for the station ID, j for the contention cycle number, k^i_j for the key used as input to the station's i hash function H^i, and finally C for the table size measured in slots. Once each station has selected a place inside the hash table, the contention is performed in the same manner like in IEEE 802.11 and the station which counts down to 0 first, is allowed to send its outstanding packet (while other stations loose the contention and the cycle repeats). Alternatively, taking a snapshot of the state of the virtual hash table just after all stations have selected their places in the table, it can be said, that the station which selected the numerically smallest slot in the virtual hash table is allowed to send its packet with the delay b^i_j measured in time slots.

Another formal view at the operation of the wireless channel contention can be also taken up under above assumptions (collisions detected by absence of ACKs and all stations hearing each other, this just for simpleness) and potentially permits to implement a variety of distributed algorithms based on a wireless station coupling. Lets denote the set of $n <= N$ stations active in the contention episode j with S_j and lets represent it by the set of station backoff timers:

$$S_j = \{b^{i_1}_j, \dots b^{i_n}_j\} \tag{8.2}$$

Then the contention cycle can be defined as a mapping $\mathcal{M} : \mathcal{S} \rightarrow C$ from the set $\mathcal{S}$ of all $S_j \in \mathcal{S}$ onto the image $C = \{0, 1\}$ and defined for non-empty S_j as:

$$m_j = \begin{cases} 0 & \text{if } b^{i_k}_j = min(S_j) \wedge \forall l \neq k \;\; b^{i_k}_j \neq b^{i_l}_j \\ 1 & \text{otherwise} \end{cases} \tag{8.3}$$

where m_j stands for the value of $\mathcal{M}$ in the contention cycle j and means $m = 0$ for successful transmission[2] and $m = 1$ for channel collision. Therefore $\mathcal{M}$ represents an elementary coupling between contending stations and their backoff timers and its value can be computed locally ((locality)) by any participating station in each contention cycle. This formal definition of $\mathcal{M}$ allows to design a broad spectrum of distributed channel algorithms based solely on the value of m_j in analogy to e.g. sorting which requires a comparison operation "<=", for example such minimising the expectation value $< m_j >$ (that is minimising the average collision rates in equilibrium) for $j \rightarrow \infty$. For a noisy channel the expression for m_j must by augmented by a discrete random variate[3] named $\sigma_j \ : \ j \rightarrow \{0, 1\}$ with an expectation value between 0 (no wireless errors) and 1 (100% loss). Depending on how the value of m_j is used in each contention cycle (whether the information contained in m_j is honored or not) a gracefull transition between a temporal sorted system (distributed sorting) and a probabilistic system (distributed hashing) is possible.

If the scheme is not distributed (which is usually not the case for wireless), each station can use the same hash function in each contention cycle, that is:

$$H^i = H^j, \ \forall i, j \tag{8.4}$$

and the access scheme can be denoted as "optimal", if the hash function H is a perfect hash function with respect to k^i_j, or in other words is a "random oracle" for any distinct sequence of input keys. Thinking this way one can imagine that each station uses its unique ID (e.g. its MAC) and the contention cycle number (which is just a counter) to generate a new backoff slot in each contention cycle by using a parametrised hash function $H_{seed=ID}(i)$[4]. Note that in this simplified access scheme real radio channel collisions can happen only, if two stations select the same smallest contention-winning table slot, but do manifestate on the radio channel (in opposite to genuine IEEE 802.11[5]), if they happen in table slots which do not win the actual contention cycle, because another numerically lower slot is the winning one. In other words: in the simplified scheme only the density of collision probability at the winning slot is of practical importance.

In a distributed access scheme like a real wireless environment there is the problem of choosing the per station hash functions H^i. But if the same parametrisable and "perfect" hash function $H_{seed=ID}(i)$ is used by each wireless station, the resulting global hash function is also going to be "perfect" with respect to the selection of the set of backoff intervals $b^1_j \dots b^N_j$ in each contention cycle, since multiplexing of N random sequences yields a random sequence too.

Such an oversimplified channel access algorithm with an uniform random distribution of backoff slots (called just $\mathcal{H}$ from now on) simplifies further studies of other collision avoidance schemes and can act as a reference model as follows. For each reasonable table size C and increasing number N of contending stations the $\mathcal{H}$-

[2]note that $m_j = 0$ means also for the contention winner numbered with l (that is the station which computed $m_j = 0$) that $b^{i_l}_j < b^{i_k}_j \ \forall k$.

[3]or a variate resembling the error behaviour of the wireless channel (e.g. bursts).

[4]That is in fact a seedable PRNG, for `pseudo-random number generator`.

[5]that is in IEEE 802.11 collisions may indeed happen logically far before they become visible on the radio channel!

scheme has been simulated[6] and the average collision probability $p_c(C,N)$ as well as the average number of empty slots (wait time) before a successful transmission $W_T(C,N)$ and before a (visible) collision $W_C(C,N)$ has been gathered (but it is possible to obtain this data analytically by calculating all possible distributions of stations over the hash table and summing up all visible collisions, e.g. C=3, N=2: [. 1 2], [1 . 2], [1 2 .], [12 . .], [. 12 .], [. . 12], ...).

If now another collision avoidance algorithm (like IEEE 802.11) is given, its statistical properties can be quantified by comparing its average $\bar{W_T}(N)$, $\bar{W_C}(N)$ and $\bar{p_c}(N)$ (obtained in another systematic simulation of that algorithm or from measurements) with values from the simple hash table scheme at the point, where $\bar{p_c}(N) = p_c(C,N)$ for a fixed number of contending stations N[7]. Simultaneously an effective window size $\bar{C}$ of the algorithm under consideration can be obtained by taking the respective C value of the $\mathcal{H}$-scheme[8]. This is a definition "by the outcome" and should not be confused with the analytical average window size as sometimes calculated for algorithms like IEEE 802.11 in the literature. While an arithmetic average window size is just a mathematical construction and in fact doesn't really provide any useful information[9], the definition of effective window size "by the outcome" in a reference system provides a direct feeling for the algorithm's performance.

I define an arbitrary collision avoidance algorithm $\mathcal{A}$ to be "better than $\mathcal{H}$", if for equal collision probabilities in both methods $\bar{p_c}(N) = p_c(C,N)$ its $\bar{W_T}(N)$ and $\bar{W_C}(N)$ values are lower than the corresponding $\mathcal{H}$-values, or to be "worse than $\mathcal{H}$", if the opposite holds. Similarly in the first case a CA-algorithm $\mathcal{A}$ can be called "channel cooperative", since for equal collision probabilities as in $\mathcal{H}$, which do not deploy any collision resolution mechanism, in the $\mathcal{A}$ scheme collisions (as well as transmissions) are handled with higher efficiency (less channel time wasted). While in $\mathcal{H}$ lowering the packet wait delay for a given number of nodes N would require a smaller table size and therefore higher collision rates, in $\mathcal{A}$ the average delay is less than in the corresponding $\mathcal{H}$ system.

To provide another practical argument, why a reference model is necessary and useful in order to compare collision avoidance schemes, I want to give a simple example. Imagine one takes 802.11 and measures or simulates its statistical properties and for example obtains a table like "collision probability is X for N contending nodes" and "average transmission wait is W for N nodes". Then a modified 802.11 algorithm called $\mathcal{A}$ with a changed minimal contention window size (e.g. doubled with respect to IEEE 802.11) undergoes the same investigation procedure and a similar table is obtained, like "in $\mathcal{A}$ collision probability for N contending nodes is $Y < X$" and "the packet wait is higher", etc. Obviously this kind of reasoning compares different things in the same manner like comparing the speed and acceleration of two cars,

[6]I have simulated the scheme for increasing number of nodes N beginning with one up to 256 nodes changing the table size C from N to $N+256$.

[7]I further assume that all of the stations are always backlogged.

[8]in practice it is not possible to find a $p_c(C,N)$ really equal to $\bar{p_c}(N)$ since C and N are discrete variables, but the value of $\bar{p_c}(N)$ can be bounded in a rectangular region of the (N,C)-area and then approximated for example.

[9]note that since 802.11 uses an exponential backoff algorithm there are sometimes very large contention window sizes used by a station for backoff generation and arithmetic averaging leads to artificially large values of the window average.

one in first gear but the second in third. But in reality the two algorithms may have the same efficiency with respect to collision avoidance and can only be compared if a common reference system is defined.

The $\mathcal{H}$ model applies also to non-probabilistic access schemes like TCF, in that case the hash function of wireless station must be set to:

$$H(i,j) \rightarrow (i+j)\, mod\, N,\ i = 1 \ldots N,\ j = 1 \ldots \infty$$

and the TCF access scheme can be simulated and quantified in $\mathcal{H}$.

8.1.2 Improved H-table Scheme

While the above simple H-scheme is well suitable for comparison purposes between different access methods, another question may be asked in the same context, which is whether the hash table approach can be further refined to provide improved performance in the sense of a wireless channel access algorithm. While a simple hash table with an uniform distribution of backoff slots provides minimal collision rates with respect to the complete table, a channel access algorithm must solve a trade-off problem between low visible collision rates and low packet wait before each transmission respective collision event.

There is also the second aspect of the collision problem with respect to channel access: while in the classic hashing problem a collision of two nodes may contribute to the overall cost regardless of where it happens in the table, if considered in the context of channel access, only collisions at the minimal table slot in each cycle influence the accountable cost. Also in perfect hashing triple or higher order collisions may impose an increased cost with respect to simple collisions but in the context of channel access the two costs are equal, since the amount of the wasted radio time in a collision does not depend on the number of colliding packets, but only on the duration of the longest packet. Nevertheless triple or higher order collisions may have a significant influence on the overall performance of a collision avoidance scheme, if collisions are viewed as information exchange between wireless stations.

In IEEE 802.11 DCF collision events are used in a non-optimal way, since in theory collisions may carry information about the internal state of other nodes (but on a high error rate link care must be taken because wireless transmission errors are in most of the cases non-distinguishable from real packet collisions). Unfortunately in an access scheme such as 802.11 it is not possible to infer anything useful about the state of other colliding partners due to the residual backoff algorithm implemented in IEEE 802.11. This is for the simple reason that in 802.11 backoff slot generation and the wait time before transmission attempts are not really related to each other. That is, while a node may choose a backoff slot b once it is going to contend for the channel, the residual wait time which finally remains in the cycle it counts the timer down to zero, is the backoff time b (in slots) minus the number of residual times of other nodes which have transmitted (or collided) while the given node was counting (and frozen on busy channel condition) from b down to zero. In other words, due to the residual backoff mechanism the packet wait time W before transmission attempts does not directly carry information about the original backoff slot selection b of the transmitting node and thus its internal state.

In the $\mathcal{H}$ scheme this is not the case and after a collision event all involved stations may infer knowledge about the state of the PRNG (or anything else used for the slot generation) of the other stations, that is to have yielded exactly the same backoff value b as the own backoff slot. This observation can be used to implement a sort of collision resolution algorithm which may lower future collision rates, once enough collisions have been observed and permitted to gather sufficient state information. This is however different from the mechanisms used for classical hash tables, because there collision resolution takes place each time new key must be inserted into one of the slots and the status of the slots (free or used) is directly available. I did not manage to design a working algorithm for wireless collision resolution yet, but the general idea is to construct orthogonal hash codes for each of the stations, taking the inferred state after collision events into account. Such an algorithm must obey to several rules:

- it can use the transmission and collision wait times in each contention cycle a given station is participating in, to deduce the state of the other stations
- it must be robust against transmission errors which could falsely mimic packet collisions
- it must cope with changing load from wireless stations, that is, maintain sufficient orthogonality if a station is not always backlogged, at least for short time periods (e.g. few contention cycles)
- it should also adapt to a changing number of wireless stations
- and finally, in the optimal case should be able to interoperate with some percentage of legacy 802.11 or Idle Sense stations and still yield better performance than a pure 802.11 network, or in the worst case, just degrade to the performance of the other algorithm

I made some experiments with a code based on permutations of backoff value sequences, where each node was using a permutation of backoff values in the range of the size of the contention window and applying a resolution scheme based on re-permutation of the respective sequences after collisions. However the algorithm yielded high amount of unfairness since it sometimes converged to a state, where some of the nodes obtained much higher fraction of the channel capacity than the others (trivial example of such orthogonal sequences is $node_1$: [1234], $node_2$: [2341] giving to the $node_1$ 75% of the total channel time, but only 25% to the $node_2$). Certainly this interesting subject requires more research. It is also worth to note that in an access algorithm with a collision resolution scheme, triple or higher order collisions may increase its convergence speed, since those higher-order collisions permit faster state exchange between stations compared to collisions of just two nodes.

Coming back to the question of a more realistic improvement of the simple H scheme in the sense of the delay-collision trade-off mentioned at the very beginning, it can be stated that each channel contention cycle can be understood as placing each of the nodes in one of the discrete slots from the contention window $W = [0, ..., C-1]$

using some, yet unknown, probability distribution[10]:

$$p_i,\ i = 0 \dots C - 1 \tag{8.5}$$

and giving the node with the smallest slot number the right to send its packet. In other words, if an external observer would always make a snapshot of the contention window (which I call the virtual hash table) at the very beginning of each contention period (before the participants start to count down their backoffs), he would observe that each wireless node is located in one of the available time slots and the one with the smallest slot wins the contention cycle (or a collision occurs)[11].

Therefore the goal of such statistical approach is to find the best probability distribution p_i in the sense of minimising the channel collision rates and the average packet transmission (respective collision) wait. To start with an intuitive approach I consider the discrete slotted case for two nodes first, however, the analysis is much easier for a continuous probability distribution $p(i)$.

Let $p_i, i \in W$ be the probability that the slot i is chosen by a node each time it must select a new backoff value. Obviously, if only two nodes have to be put into the C discrete table slots, the per-slot probability that they collide at the slot number i can be written as:

$$P_{col}(i) = p_i \cdot p_i \tag{8.6}$$

and since node insertions are statistically independent, it is just the probability that $node_1$ is in slot i multiplied with the probability that the second node is in the same table slot too. On the other hand, if the two nodes do not collide during an insertion cycle, the probability that one of the nodes selected the table slot numbered with i, but the second selected a higher table slot (that is the transmission probability at the slot i) can be written as:

$$P_{trans}(i) = 2 \cdot p_i \cdot \sum_{j>i}^{C-1} p_j \tag{8.7}$$

where the factor of two in the front of the equation comes from the distinguishability of the two nodes[12]. Further I want to introduce a cost function ζ, which represents the cost imposed by using the hashing procedure as a radio access protocol:

$$\zeta = \sum_{i=0}^{C-1} (A(i) \cdot P_{col}(i) + B(i) \cdot P_{trans}(i)) \tag{8.8}$$

where the factor $A(i)$ represents the cost of a channel collision, if the two nodes selected table slot number i and collided in the following packet transmission attempt, and analogically $B(i)$ represents the cost of a transmission, if one of the nodes selected table slot i and then successfully transmitted its packet. In other words, the cost expressed by ζ represents the wasted radio channel time and is subject to minimisation.

[10]**that is p_i is the probability that a participating station will select table slot number i.**

[11]**In language of statistics: a Bernoulli experiment.**

[12]**Indistinguishable objects appear up to my knowledge only in quantum world.**

To find suitable expressions for the A and B factors, it can be observed that a collision event means a lost packet, thus it means wasted channel time, while on the other hand in a successful transmission event channel time is well used to transmit some data. Therefore the cost A for a collision for a packet of the size s can be defined to be the time spent in waiting before a send attempt can be done and the time wasted during sending the packet. In contrast the cost B of a transmission is only due to the contention wait before the packet gets on the radio channel. This consideration leads to the following relations for the two cost factors A and B at the given table slot i counting the system time in units of a contention slot:

$$A = i + T(s) \tag{8.9}$$

$$B = i \tag{8.10}$$

where $T(s)$ stands for the time lost during the sending of the packet of s bytes in size over the air (which means e.g. for 802.11 that the corresponding IFS intervals and the duration of the MAC level ACK must be added to the physical transmission time of the data frame). Further analysis is easier, if the discrete slot system is replaced by a continuous one, that is, let $p(i)$ be the probability distribution function (PDF) over the normalised contention interval $W = [0, 1]$ sampled every time stations are placed into slots during the contention phase[13]. Further let $c(t)$ be the cumulative probability distribution function (CDF) derived from $p(t)$, that is:

$$\int_0^1 p(t)dt = 1 \tag{8.11}$$

$$c(t) = \int_0^t p(t')dt' \tag{8.12}$$

After an analogous consideration as for the discrete case, the corresponding probabilities (or more precisely probability densities) for collision and transmission events in a continuous system can be expressed by following relations with respect to the unknown probability distribution p:

$$P_{col}(t) = p(t) \cdot p(t) \tag{8.13}$$

$$P_{trans}(t) = 2 \cdot p(t) \cdot \int_t^1 p(t')dt' = 2 \cdot p(t) \cdot (1 - \int_0^t p(t')dt') = 2 \cdot p(t) \cdot (1 - c(t)) \tag{8.14}$$

While the first expression is rather clear, the second of the two equations above can be interpreted as having one of the nodes at the position i in the normalised contention window while having the second one anywhere above in the normalised window (note that for a bounded function like a PDF[14] the integral over the half open interval $(i, 1]$ equals to the integral over $[i, 1]$. In the same manner as for the discrete case a cost functional (which is simply speaking a mapping from a function

[13]But discrete system can be reobtained from the continuous one by integrating the PDF over each of C subintervals mapped onto the C discrete slots.

[14]while it is possible to have singularities (δ-function) in a probability distribution, I look here only for continuous ones, since the PDF is intended to generalise the discrete slot system where I assume, that the per-slot probabilities are not zero.

space, usually a Banach space to real or complex numbers) representing the wasted channel time can be introduced and expressed as:

$$\zeta[p] = \int_0^1 (A(t) \cdot P_{col}(t) + B(t) \cdot P_{trans}(t))dt = \int_0^1 F(p(t), c(t), t) \tag{8.15}$$

which is a functional of the sought probability distribution $p(t)$ and its cumulative distribution function $c(t)$.

A necessary condition for the cost functional to take an extremal value is, that its integrand must satisfy the Euler-Lagrange differential equation[15], which holds for any F of the form $F(c, p = \frac{dc(t)}{dt}, t)$:

$$\frac{\partial F}{\partial c} - \frac{d}{dt}\frac{\partial F}{\partial p} = 0 \tag{8.16}$$

and if applied to the cost expression for the sought PDF/CDF yields after substituting the relations for $A(t)$ and $B(t)$ from above:

$$-2\,(\frac{\partial}{\partial t}\,\mathrm{c}(t)) - 2\,(\frac{\partial^2}{\partial t^2}\,\mathrm{c}(t))\,t - 2\,(\frac{\partial^2}{\partial t^2}\,\mathrm{c}(t))\,T(s) - 2 + 2\,\mathrm{c}(t) = 0 \tag{8.17}$$

The equation can be solved analytically but yields a complicated expression in-

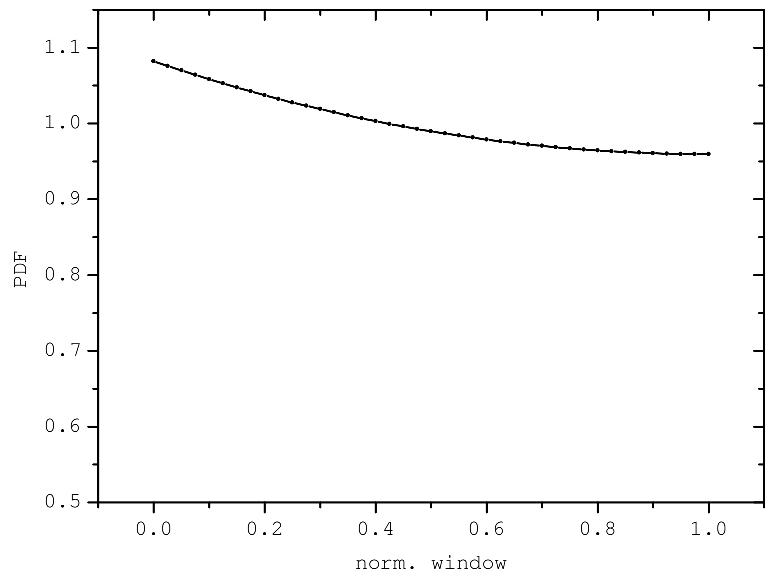

Figure 8.1: Approximated optimal PDF for $T(s) = 4$ in the simplified 2 nodes case.

volving complex Bessel functions of first and second kind. Further simplification to

[15]**Which can be easily derived by computing the first order variation of the cost integral $\delta\zeta$ and partial integration of the resulting expression taking into account that the variation vanishes at integral boundaries and is a similar extremity condition for functionals like $\frac{df(x)}{dx} = 0$ for "normal" functions.**

the structure of the above equation can be performed assuming that collisions are relatively rare events and much more costly than the average time spent in waiting before a collision can be detected. Therefore assuming that the collision cost A is a constant over the definition domain and depends only on $T(s)$ yields[16]:

$$-T(S)\,(\frac{\partial^2}{\partial t^2}\,\mathrm{c}(t)) + \mathrm{c}(t) - 1 = 0 \tag{8.18}$$

which after some mechanical computation yields now a simplified expression for the CDF $c(t)$:

$$\mathrm{c}(t) = 1 + \frac{\cosh(\frac{1}{\sqrt{T(s)}})\sinh(\frac{t}{\sqrt{T(s)}})}{\sinh(\frac{1}{\sqrt{T(s)}})} - \cosh(\frac{t}{\sqrt{T(s)}}) \tag{8.19}$$

after taking the obligatory boundary conditions for $c(t) : c(0) = 0, c(1) = 1$ (normalisation of the probability distribution) into account. The corresponding probability distribution function can be obtained by computing the derivative of the CDF over t and is for a reasonable range of collision costs (e.g. collision cost 4 times more than the duration of the contention interval) close to a straight line with a negative slope over the normalised window interval $[0, 1]$ as shown in the figure 8.1.

The intuition behind this result is as follows: while to only avoid collisions it is optimal to distribute the stations over all of the available M table slots with equal probability $p = \frac{1}{M}$, it is more advantageous to use slots at the beginning of the table to decrease the packet wait delay and optimise the total cost with respect to wireless channel access. The trade-off between the two contrary goals yields the form of the distribution function as depicted above.

These considerations can be extended to the generalised N nodes case and an analogous expression for the integrand of the cost functional can be written as:

$$\begin{aligned} F =&(A+t)\cdot\left(\binom{N}{0}p(t)^N + \binom{N}{1}p(t)^{N-1}(1-c(t)) + \ldots + \right.\\ &\left.\binom{N}{N-2}p(t)^2(1-c(t))^{N-2}\right) + t\cdot\binom{N}{N-1}p(t)(1-c(t))^{N-1} \\ =&(A+t)\cdot\sum_{i=0}^{N-2}\binom{N}{i+1}p(t)^{N-i}(1-c(t))^i + tNp(t)(1-c(t))^{N-1} \end{aligned} \tag{8.20}$$

where the consecutive terms can be interpreted as contributions of the different collision orders, that is of the collisions of two, three, etc. nodes at the minimal table slot to the overall cost, and the last term contributing for the successful packet transmission cost. If inserted into the Euler-Lagrange equation the expression yields very complex differential equations for the optimal integral probability distribution $c(t)$, too complex for me to find an analytical solution and even if it could be solved

[16]**Note that I do not simplify the differential equation but put the simplified expression for A into the full equation above and this is perfectly legal.**

analytically, the solution will be different for each number of contending nodes N[17]. However, I think that after solving the simplified two node case, the structure of an optimised density function with respect to channel access and wasted channel time can be similarly argued for. For N nodes it is also more advantageous to select lower table slots with slightly higher probability than numerically bigger slots in order to minimise packet wait delay, but if the probability to choose lower slots is too high, too many collisions will occur and increase the total cost.

At this point it is also worth to note that the collision and transmission terms[18] in the above equation for the cost integrand can be used to analytically compute the statistical properties of any $\mathcal{H}$ system, once its probability distribution $p(t)$ is given. This simplifies the analysis of an arbitrary $\mathcal{H}$, because a time consuming simulation is not necessary anymore.

Paradoxically, if the effective distribution of IEEE 802.11[19] is sampled, a similar result, that is a ramp down from some maximum value to a minimum value can be observed[20]. One may argue that 802.11 is not a random process in the sense of the H-table construction because stations do not reselect backoff intervals each cycle but freeze their timers and continue to count each time the channel becomes free. In fact, 802.11's residual backoff scheme is equivalent to a random process with a residual window size, where at the end of each contention cycle stations do not freeze their backoff timers but reduce their respective contention window used to select a random backoff in the next cycle by the amount of slots they have waited in the actual cycle and always draw a new random slot in the following cycle. I simulated this modified scheme (not shown here) and it yields exactly the same statistical properties (collision rate, packet wait) as the original 802.11 algorithm (of course, the sequence of states for each stations differs from 802.11, and therefore the short-term fairness too). Hence it is legal to speak about an effective probability distribution of IEEE 802.11.

8.2 Simulation Results

While the previous section provided a rationale for replacing the access method of 802.11 by a generalised probability-distribution approach, this section provides and discusses simulation results using the insights about collision avoidance derived before. Although a general solution for an optimal probability distribution in the case of N contending nodes could not be found in an analytical way, a more brute-force approach is now chosen taking into account what has been said about the trade-off between collisions and average packet wait.

For simulation purposes a simple finite state automaton model of the 802.11 DCF access method has been coded in C. Only the channel contention phase is simulated

[17]A possible way out of this problem for future research may be the use of Monte-Carlo simulations or Simplex-optimisation with a parametrised $p(i)$ to find an approximation for improved probability distribution p.

[18]If transformed back into a discrete slot system the CDF $c(t)$ must be replaced by the respective probability sum.

[19]That is, distribution of backoff intervals at the beginning of each contention cycle.

[20]Which is piecewise continuous due to the exponential backoff algorithm.

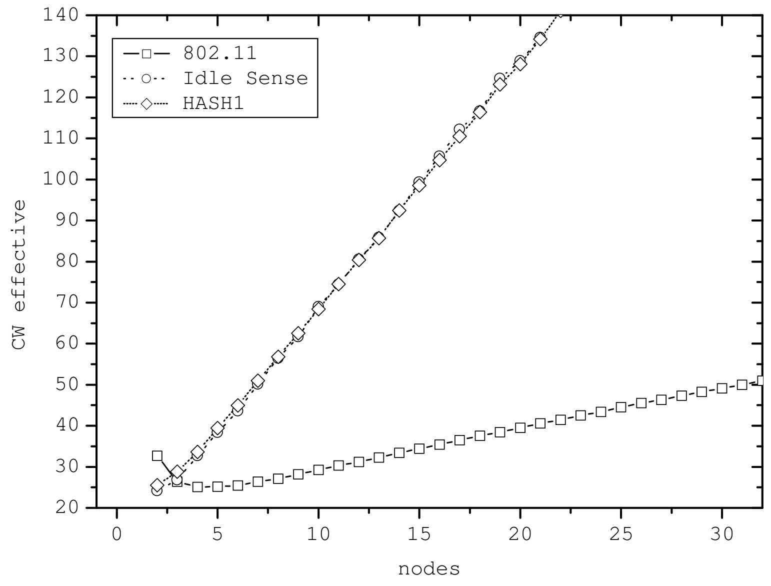

Figure 8.2: Effective contention window sizes for 802.11, Idle Sense and HASH1 access methods.

here, but not all of the details which are present in a real 802.11 (that is for example packet retransmission counters, dynamically changing station loads, etc.). The simulation code permits to select the type of the backoff algorithm used (residual backoff on/off, exponential backoff on/off after collision) and takes an arbitrary probability distribution function $p(t)$, which is used to generate backoff slots. If $p(t)$ is set to a constant function over the definition domain and both backoff mechanisms are enabled, the original contention scheme of 802.11 is obtained. If exponential backoff is disabled and correct contention window values are provided to the code, the statistical (average) behaviour of `Idle Sense` can be studied[21]. Otherwise with the two backoff mechanisms off and an arbitrary distribution $p(t)$, a general H-table scheme can be studied.

For random univariate generation the `UNURAN` package [64],[86] has been used. The `UNURAN` library requires the specification of the probability distribution function PDF and for some of generation methods, either the derivative of the PDF or the corresponding CDF must be provided. If CDF is not required, the PDF doesn't need to be normalised.

To study the trade-off idea between wait time and collision time in a generalised $\mathcal{H}$-scheme, an ad-hoc PDF of the form:

$$p(t) = 1 - t, \frac{\partial p(t)}{\partial t} = -1 \tag{8.21}$$

[21]**My simulation code does not implement the AIMD window control of Idle Sense, that is convergence dynamics cannot be studied.**

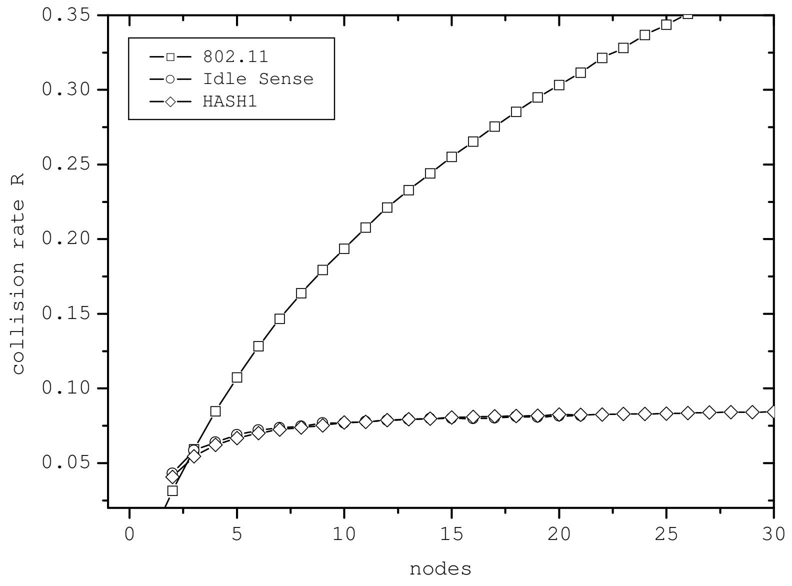

Figure 8.3: Simulated collision rates for 802.11, Idle Sense and HASH1 channel access method.

has been used for any number of wireless nodes. This is rather a radical choice but fits the requirement of giving a higher probability to lower table slots as postulated in the previous section. The corresponding scheme has been called HASH1. All simulations presented here have been ran until one million of packets has been successfully transmitted over the simulated wireless channel.

In the first simulation run both IEEE 802.11 ($CW_{min} = 32$, $CW_{max} = 1024$,) and Idle Sense (optimal contention window sizes taken from [60]) have been evaluated and their effective window sizes estimated from the comparison with the simulated two-dimensional data set for the uniform reference system $\mathcal{H}$. Here the power of a reference model becomes visible - while it won't be otherwise possible to quantify something complex like 802.11 using residual and exponential backoff, a reference model easily permits the definition of the effective window size.

The result can be viewed in the figure 8.2 and it is clear that for the regular IEEE 802.11 the effective window size drops from $N = 2$ (where all three curves begin) to $N = 4$, to slowly increase from $N = 5$ on with the growing number of contending nodes and illustrates the window anomaly mentioned while discussing the equal packet rate test in the previous chapter. In its mostly linear section from $N = 5$ on the 802.11 curve can be described by a line with a slope of about 0.9 per contending node, that is in other words, the combined effect of residual and exponential backoff in 802.11 is to add roughly 0.9 slots per contending node to the virtual hash table having the equal collision probability as the original IEEE 802.11 access scheme.

In order to permit a further comparison between HASH1 and the other two algorithms, the effective-window curve obtained from Idle Sense simulation has been used to find the appropriate contention window sizes for the HASH1 scheme yielding equal effective contention window sizes[22] (a relation for the contention window CW of $CW = A + B * N$ has been assumed and A, B varied, until the effective window sizes of Idle Sense with residual backoff and HASH1 without residual backoff equalised). This is necessary in order to compare the effect originating from the residual backoff with what can be obtained by choosing a non-uniform probability distribution $p(t)$. It is obvious from the linear part of the Idle Sense effective contention window size curve, that the combined effect of setting (converging to in reality) of the contention window to the target value (see [60] for exact target window values) can be quantified as adding of exactly 6 more slots to the virtual hash table. This is of course less than the absolute growth in the size of the contention window in Idle Sense, since the residual backoff algorithm has a counter effect on the effective window size. In HASH1 with the ad-hoc PDF (but without residual backoff) the gross window size must be grown by roughly 12 slots for each additional node, in order to obtain equal collision probabilities like in Idle Sense for each N.

The second figure 8.3 shows resulting collision rates R (where R is defined as: $R = collisions/transmissions$ and the collision probability can be obtained from $p_c = R/(R+1)$) for the three algorithms. While the effective contention window increase in IEEE 802.11 is not big enough to keep the collision rates constant, the window growth in Idle Sense exactly compensates for increasing collision rates, if more and more nodes contend for the channel. This is not a new insight, since that's the goal of the Idle Sense access method, but it is interesting to see that same compensation can be achieved for an arbitrary probability distribution function and seems to be linear for any other ad-hoc distribution too, which I quickly tested.

The two following figures 8.4 and 8.5 provide an insight into the effect of the residual backoff procedure on the packet wait common to 802.11 and Idle Sense. Before I can discuss the result, an explanation of what is presented in the figures must be given. There are three pairs of curves in each figure, each pair with same symbol shape belongs to one algorithm, rectangle is IEEE 802.11, circle Idle Sense and diamond HASH1. In each pair the curve with empty symbols shows values obtained from the respective simulation, while the filled symbol curve belongs to values obtained from the approximation based on the two-dimensional $\mathcal{H}$ data set for same collision probability and same number of contending nodes. On the X axis the number of contending nodes is given, while the Y axis shows the mean packet wait delay measured in time slots.

In case of the standard IEEE 802.11 access scheme it can be stated, that the algorithm provides lower packet wait before transmission for up to 4 contending nodes than the corresponding H-table scheme with equal collision probability. For more than 4 nodes the difference virtually vanishes. With respect to the collision wait time, 802.11 provides lower delay for up to 5 nodes but then the difference is getting less than one time slot and also vanishes quickly. This effect can be attributed to the residual backoff in 802.11, which plays a similar role as a cost-optimised probability distribution. In fact, the residual backoff scheme provides (on average) a

[22]**Therefore equal collision rates!**

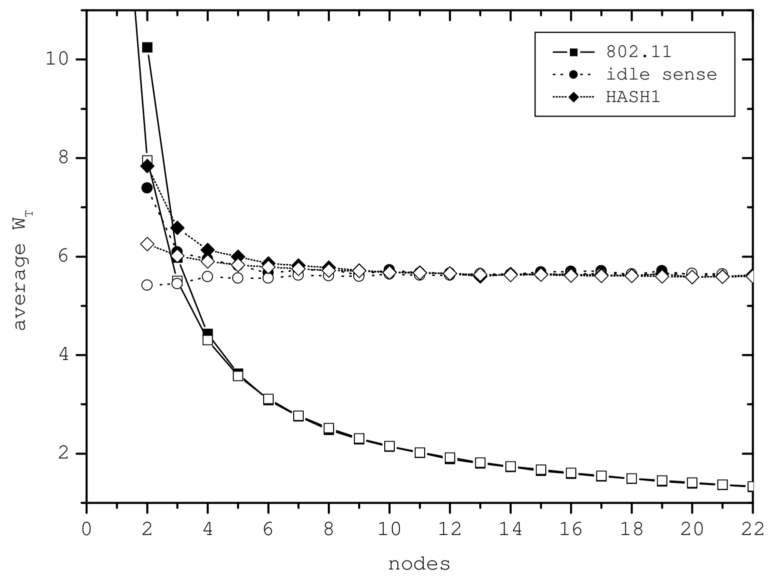

Figure 8.4: Average simulated transmission wait for 802.11, Idle Sense and HASH1.

partial temporal sorting of nodes with respect to their backoff timers. If all nodes are backlogged and, at a given time, use an equal contention window CW, a node which has waited k slots in a contention cycle and then has transmitted the packet, can assume that no other station will have a residual backoff slot in the range of $C-k,...,C-1$. That is in other words the residual backoff scheme moves nodes which did not send a packet into lower slots of the contention window.

In the direct comparison between Idle Sense and HASH1 it can be stated that both are better for very few nodes than the corresponding equal-probability $\mathcal{H}$-system, however the mechanism to establish the difference in the average wait times varies between the two algorithms. While the W_T and W_C values of the two schemes are not really equal (and this can be contributed to the ad-hoc choice of the PDF in HASH1), both schemes show mostly equal deviation from $\mathcal{H}$ counted in time slots. This can be interpreted as statistical equivalence of the residual backoff algorithm in an equal probability distribution scheme like Idle Sense with an appropriate probability distribution applied to a $\mathcal{H}$-like access scheme with respect to their statistical packet wait times.

Another important comparison criterion that must be considered here is the short-time fairness. In the established literature the Jain's fairness index [74] applied to the number of inter-transmissions between two packets sent by a particular node is widely used to quantify short-term fairness of wireless access algorithms [59]. The index is very sensitive to small variations in stations' packet rates at short time scales (short with respect to the average duration of a channel transmission cycle multiplied with the number of contending nodes).

It is well known that the original 802.11 DCF scheme as designed by IEEE exhibits

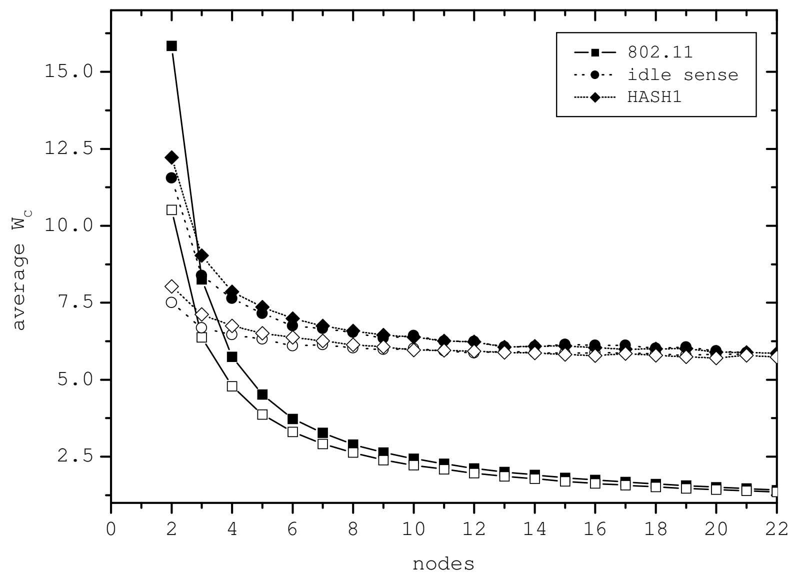

Figure 8.5: Average simulated collision wait for 802.11, Idle Sense and HASH1.

decreasing short-time fairness for growing number of contending nodes. This has been broadly attributed in the literature to the exponential backoff algorithm which induces huge variations of contention window sizes used by each node to select backoff values in consecutive channel access cycles. In other words, while some stations may have experienced a collision and have doubled (or quadrupled, etc.) their temporary contention windows, others may have sent a packet successfully the last time and have reset their contention window to the minimum 802.11 value of 32 slots. Therefore some of the stations have higher probability to select a backoff slot which is much bigger than current backoff values of other stations resulting in poor short-time fairness.

While the above reasoning is the usual point of view taken to look at the short-time fairness, I believe that the sources of fairness and unfairness must be deeper studied by analytical means in order to provide a generalised insight into the fairness problem in the context of a wireless access algorithm. To provide a starting point the fairness of both Idle Sense and the HASH1 scheme has been compared and it can be said that the short-term fairness of Idle Sense seems to be slightly better (as shown in the figure 8.6) than in an equal-collision rate HASH1 scheme, which can nevertheless be considered to be still very fair.

It is important to note that short-term fairness of HASH1 doesn't depend on the choice of the probability distribution used to draw backoff slots. This is maybe surprising at the first glance, but since HASH1 is just a random process, where no coupling exists between the contending stations, each of the stations has always the same probability to send its next outstanding packet in each contention cycle. Hence the probability of sending a packet does not depend on station's recent history, that

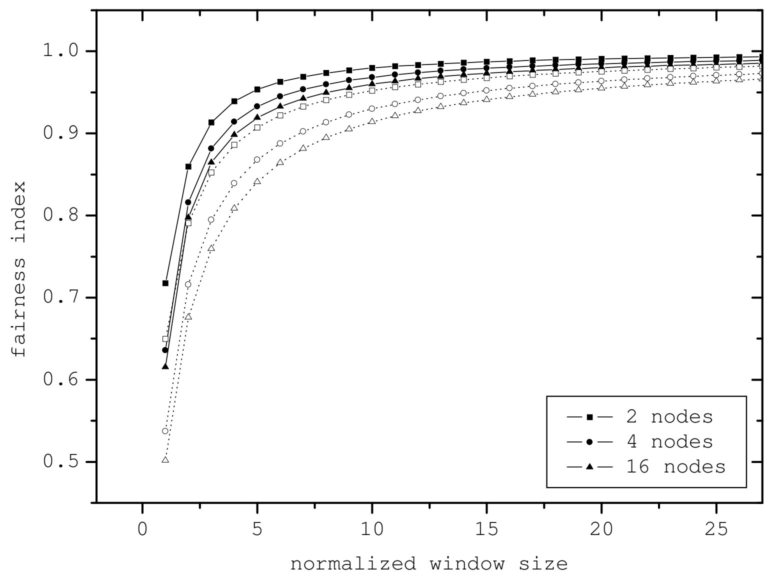

Figure 8.6: Jain's fairness index for Idle Sense (solid symbols) and HASH1.

is the amount of time which has passed since the station transmitted the preceding packet. Therefore short-term fairness of HASH1 can also be used as reference line, in order to compare different access schemes and in fact its respective Jain's fairness index can even be computed numerically without any simulation. Another important conclusion is that it is safely possible to modify the probability distribution used in HASH1 without influencing its short-term fairness while anything similar is not easily possible for IEEE 802.11 and Idle Sense, because any modification to their building blocks (exponential and residual backoff) may substantially influence their short-term fairness behaviour.

After those preliminary digressions it is time to understand the real source of fairness and unfairness in above two access algorithms. It is rather clear that studying only the Jain's fairness index does not lead to further insights into the problem of short-term fairness, therefore another, complementary statistical quantification must be introduced for further analysis. Since the fairness in HASH1 has been identified to provide a sort of a reference line, intuition suggest the introduction of a statistical metrics, which quantifies the deviation in the short-time fairness of any arbitrary access algorithm $\mathcal{A}$ from the equal cycle-by-cycle probability scheme like $\mathcal{H}$. While a number of such metrics can probably be found, I think that an useful one can be the self-correlation coefficient, respective the self-correlation probability p_s[23], which can be considered as a complementary metrics to the Jain's fairness index J.

The self-correlation probability index $p_s(t)$ can be defined in the context of a wireless channel access problem as the probability that a station, which has sent a packet at

[23]Sometimes also called the autocorrelation index.

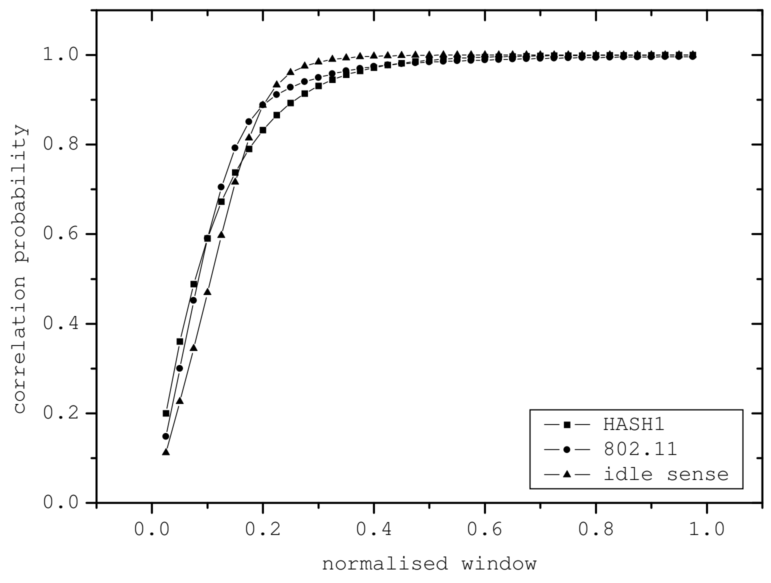

Figure 8.7: Self-correlation index for HASH1, 802.11 and Idle Sense, 5 contending stations.

the time T will have sent another packet at the time $T+t$, averaged over a sequence of channel access events[24]. In practice it makes sense to define $p_s(t)$ not over time but over a sequence of wireless transmission cycles marking each of the N nodes with an unique ID $n_i, i = 1 \ldots N$[25] and map the index to a reference interval that I have chosen to be $[0, 1]$. Note that this is numerically different from the usual definition domain of the Jain's fairness index, where the normalised window spans the range of $[0, 100]$.

If a channel access sequence of M cycles in length $S = \{n_{ik}, k = 1 \ldots M\}$ is given, the $p_s(t)$ index value can be computed as follows: for each consecutive starting event $e_k \in S$, $k = 1 \ldots M - W$ a pointer $j = k + 1 \ldots k + W$ is advanced in S over the finite window W and bin counting is performed in W, increasing the bin count in W at position j, if the node with the unique ID n_{ik} has sent another[26] packet between $k + 1$ and j. At the end of the counting procedure the bin counts are normalised in order to express the transmission probability of at least one more packet relative to the previous packet transmission. In the remaining part of this chapter the look-ahead window size W has been set to 8 times the number of nodes which seems to be reasonably big[27].

The three schemes discussed above have been now simulated for 5 and 15 stations

[24]**An alternative definition may consider not the cumulative but per-slot probability, that is, the probability that once a station has sent a packet at the time T, it will send another packet at the time $T+t$ - note the absence of the word "have"**

[25]**that is over a sequence of node transmissions like $\{14252132\ldots\}$.**

[26]**At least one more.**

[27]**Attention, the definition interval of $p_s(t)$ spans a differently sized access sequence interval depending on the number of nodes it is being calculated for.**

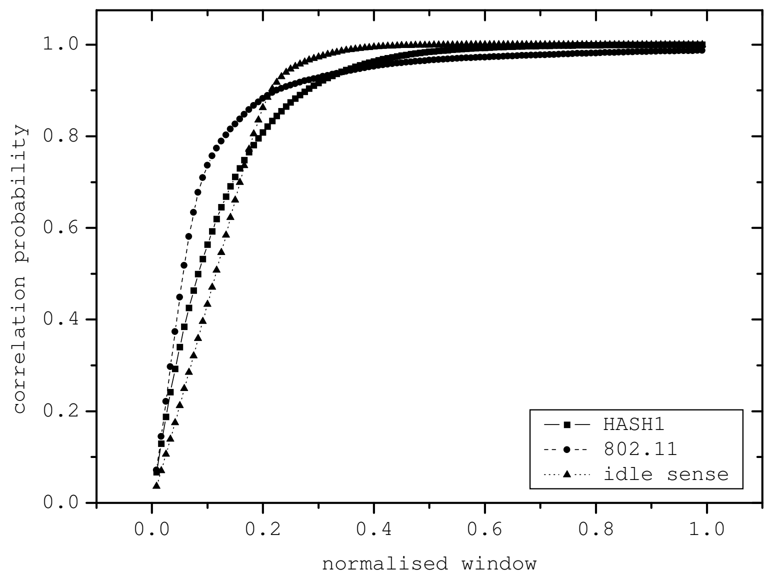

Figure 8.8: Self-correlation index for HASH1, 802.11 and Idle Sense, 15 contending nodes.

and their fairness and correlation indexes captured and shown in figures 8.8 and 8.7 respectively. Both figures clearly show, where the additional fairness of Idle Sense over HASH1 is coming from: it trades correlation for fairness[28]! I want to discuss the result in more detail now and finish with an unexpected conclusion.

Although the quantitative results for the three access schemes differ slightly for changing number of contending nodes, it can be concluded that in the Idle Sense access method with respect to HASH1, the residual backoff scheme slightly lowers the probability that a station, which has sent a packet at T will also have sent its next packet at $T+t$ (the HASH1 curve with rectangle symbols climbs faster than the triangled curve for Idle Sense for up to $W = 0.15$ in the case of 15 nodes). In IEEE's 802.11 similar effect exists only at the very beginning of the normalised window W and the curve from the 802.11 simulation crosses the HASH1 curve very quickly. In other words, residual backoff has a repulsive effect on the transmission probability of the station, which has successfully sent a packet.

A word must be said about the TCF access method, before I continue with the discussion of the above finding: in the TCF paper [87] the short-term fairness of the TCF scheme is found to be very high, even higher than in Idle Sense. If one would calculate the self-correlation coefficient of TCF, the corresponding curve would consist of a sharp step[29] from 0 to 1 at the point, where the TCF cycle repeats, that is in case of setting $W = 8$ for $p_s(t)$ calculation, at about $1/8 = 0.125$. This happens since in TCF each node has a fixed place in the TCF super-cycle, which guarantees each of the nodes the transmission of one more packet per super-cycle. Here again,

[28]Which is per se neither good nor bad.

[29]In steady state with a constant number of always backlogged stations.

computing (or imagining) the correlation index provides an useful tool for gaining insights into the source of the increased fairness of TCF.

Generalisation of above findings is the insight, that there is a natural limit for the level of short-term fairness marked by the fairness curves of any equal-probability scheme like any $\mathcal{H}$ scheme and governed by the corresponding correlation probabilities. Deviations (in any direction) of an access algorithm $\mathcal{A}$ from the reference fairness of $\mathcal{H}$ can be quantified by its $p_s(t)$ index and attributed to the deviation of the self-correlation coefficient of $\mathcal{A}$ from the reference line defined by that of $\mathcal{H}$. Exponential and residual backoff however, are both mechanisms, which can in principle influence the fairness of the incorporating access scheme in any direction and cannot be separated from each other. Fairness, at least with respect to the Jain's fairness index, can be improved, if a station which has sent a packet, is for a while given a lower probability to send its next packet than stations which did not send a packet and this is exactly the opposite thing compared to what happens in the original IEEE 802.11 design.

Since it seems so obvious that the fairness problem in 802.11 is related to the in-

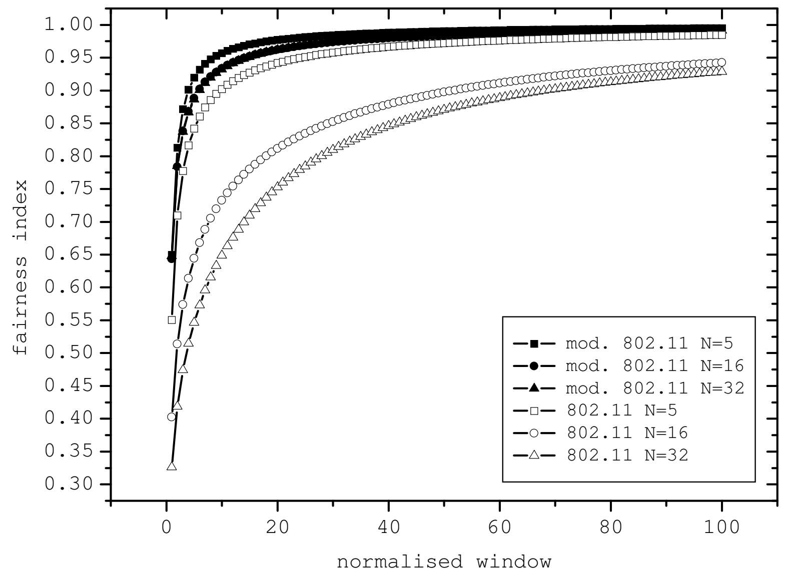

Figure 8.9: Jain's fairness index for 802.11 and the modified version 802.11-M.

creased sending probability of stations which have recently sent a packet with respect to those which did not, an ad-hoc modification (802.11-M) is now introduced into the 802.11 channel contention algorithm to verify the sensitivity of the short-term fairness to small changes in the sending probabilities of stations, which have recently sent a packet. The modification is very simple and only requires a station wanting to generate a new backoff slot to choose the backoff value not between 0 and $CW-1$ (its actual contention window, including doubled windows after collisions etc.), but to choose the value rather between M and $M+CW$. This construction permits to lower the immediate sending probability of a station which has recently performed a transmission.

The result of the modification can be examined in the figure 8.9, where the M value

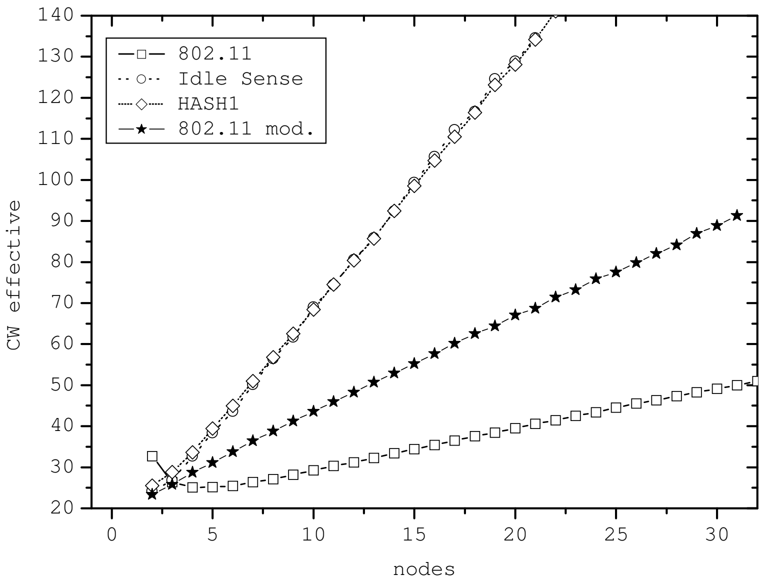

Figure 8.10: Effective contention window size of 802.11-M.

has been just set to two times the number of contending stations, that is the backoff slot selection each time a new value must be drawn has been modified to:

$$B \rightarrow Random()\, mod\, CW \; + \; M$$

where CW stands for the temporary contention window size at the time the backoff slot is selected. The choice of setting M to two times the total number of stations is just an ad-hoc choice for simulation purposes and not intended to be performed in a real-world algorithm, where stations do not know the number of contending MAC units in advance. However, since in 802.11 the collision rates experienced by a host depend only on the number of contending stations, it is imaginable to introduce a correction factor based on the actual channel collision rates.

The result shows clearly that the short-term fairness issue in wireless networks is not directly related to the exact structure of the backoff algorithm performed after a collision, but only indirectly, by correlation between the transmission probabilities for a given node at a reference time T and a later time $T+t$. As directly visible from the figure 8.9 the short-term fairness of the modified 802.11 scheme and quantified by the Jain's fairness index is very close to the result presented for Idle Sense (figure 8.6 but also see the Idle Sense paper [60]), although the high level of fairness in the modified 802.11 scheme has not been just obtained through an equal growth in the effective contention window size as can be observed in the figure 8.10, where the already presented curve has been augmented by the curve for the modified 802.11

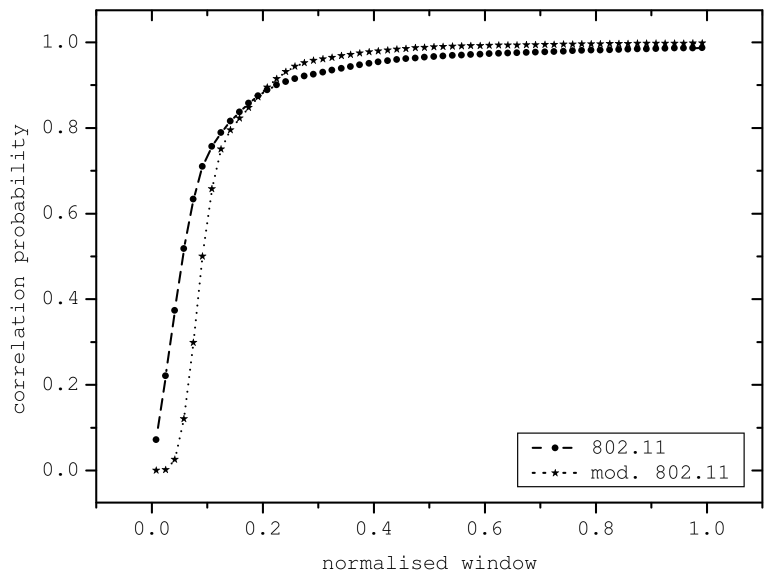

Figure 8.11: Self-correlation index for 802.11 and modified 802.11 for 15 contending nodes.

scheme. Finally, fig. 8.11 shows that improvements in fairness are, as predicted, accompanied by corresponding changes in the self-correlation index.

There is indeed an additional growth in the effective contention window size with respect to the original IEEE DCF 802.11 scheme, but it is far below the curve obtained for Idle Sense. The collision rates for the modified 802.11 algorithm remain very high with respect to Idle Sense, but are slightly lower than in original 802.11 and only this issue can be attributed to the increased effective window size. The conclusion is that in a generalised CSMA/CA wireless protocol collision rates and short-term fairness are independent phenomena and can be controlled separately, the first by congestion control applied to sending probabilities of contending stations at a given time t, the second by performing suitable correlation control of sending probabilities of the same node but at different times.

The result permits to revide the definition of the short-term fairness. While the Jain's fairness index J has been the preferred way to quantify the short-term fairness of any access algorithm for a while, in my opinion there isn't just a fairness scale "unfair, more fair, most fair" like suggested by J, but rather "negatively correlated, uncorrelated, positively correlated" with the fairness and correlation curves of any $\mathcal{H}$ acting as reference line and origin for defining the short-time fairness if accessing a wireless channel.

8.3 Conclusion and Future Research

In this section I have provided a statistical method capable of comparing different collision avoidance schemes and have given arguments for replacing the residual backoff mechanism in any 802.11 like scheme by an uniform random process using an optimised (and in general, continuous) probability distribution $p(t)$ over a normalised contention window[30]. Once residual backoff has been eliminated a door is open for implementing more complex collision resolution schemes which use collision signals and information exchange between the MACs as described in the section preliminaries. Furthermore, the real source of short-term unfairness has been identified as the positive time-correlation between consecutive packet transmissions of a node and a quantification metrics ("control knob") in form of the autocorrelation coefficient has been presented. This opens a new direction, where the future research of CSMA/CA type algorithms may go.

Another important future research direction arises from taking a look at the work going on in the area of MAC level QoS, namely the IEEE 802.11e Working Group [67]. In IEEE 802.11e the basic DCF access scheme is extended to provide a number (current IEEE draft permits up to 4) of virtual MACs (so called access categories, ACs) with configurable DCF parameters in the physical MAC unit of a wireless station (an extension to DCF called EDCF). The three-priority (SIFS, PIFS, DIFS) inter-frame space scheme of 802.11 is augmented to provide an individual so called AIFS (arbitration inter frame space which should be DIFS plus some, possibly zero time) to each of the virtual MACs together with configurable minimal and maximal values for the contention window size. Each frame in a 802.11e capable station must be marked by upper layers (e.g. IP) with a specific priority value in the range of 0 to 7, which is mapped to one of the available ACs. In the infrastructure mode IEEE 802.11e permits the cell coordinator to announce AC parameters in the beacon frames, in order to adapt the EDCF parameters on the mobiles to changing traffic conditions. Appropriate setting of the AC parameters is capable of providing MAC service differentiation between the traffic classes. Additionally, a packet bursting mode is available in 802.11e, in order to enhance the performance and achieve better medium utilisation[31]. Numerous literature ([88], [32], [119], [101]) exists and evaluates the performance of the IEEE 802.11e scheme in a number of scenarios (by simulation since real 802.11e products became available very recently), however, the common conclusion is that it is difficult to find suitable values for the virtual MAC parameters in order to simultaneously satisfy all of the imposed QoS requirements. The goal of the effort made in the IEEE 802.11e Working Group goes more into the direction of providing a suitable but simplistic support for a data and voice-over-IP solution for home users. Recently major Internet providers (like Deutsche Telekom in Germany or Wanadoo in France) have launched a broad campaign offering cheap flatrates for Internet over ADSL together with unlimited IP-telephony, usually bundled with 802.11 like wireless equipment for home use and this requires exactly what 802.11e claims to provide.

While the ongoing work in 802.11e seems to deliver at least a potentially generic mechanism capable of providing traffic differentiation at the MAC layer, it is go-

[30]**While a simulation may with easiness use an available random-number library like UNURAN, a practical in-hardware implementation may encounter complexity problems while dealing with arbitrary $p(t)$.**

[31]**Since upcoming products provide 54Mbit (802.11g) or even 108Mbit radio units, the contention time starts to exceed the physical packet transmission time, if done on a per-packet basis being therefore very ineffective without a bursting mode.**

ing to suffer from similar problems as standard 802.11, namely high collision rates for more than just very few nodes in the cell, short-term unfairness, performance anomaly [58], etc. Unfortunately it is not easily possible to extend the dynamic contention window control of the Idle Sense access method to a multiple virtual MAC scheme like IEEE 802.11e. While the AIMD control deployed in Idle Sense leads to the convergence of contention window sizes among participating stations, the traffic differentiation mechanism in 802.11e depends on the use of different contention window sizes as well as different interframe spaces for transmiting packets in each of the configured classes.

It could be however possible to apply an AIMD like control to 802.11e EDCF window sizes in a simple case, for example assuming that there are just two classes with completely distinct MAC-level priorities[32], that is if the high-priority class always preempts the second low-priority class. In such scenario it is imaginable to apply AIMD control separately to each of the classes and dynamically adapt MAC parameters in order to 1) keep collision rates low for each class, and 2) to maintain relative priorities between the two classes. However, in a general situation the virtual MACs will have overlapping contention intervals [32] and here an AIMD approach is not going to work as intended, since it can not be certainly anymore told, in which of the two classes the channel congestion actually happens.

One may argue that once a wireless MAC access algorithm like Idle Sense is widely

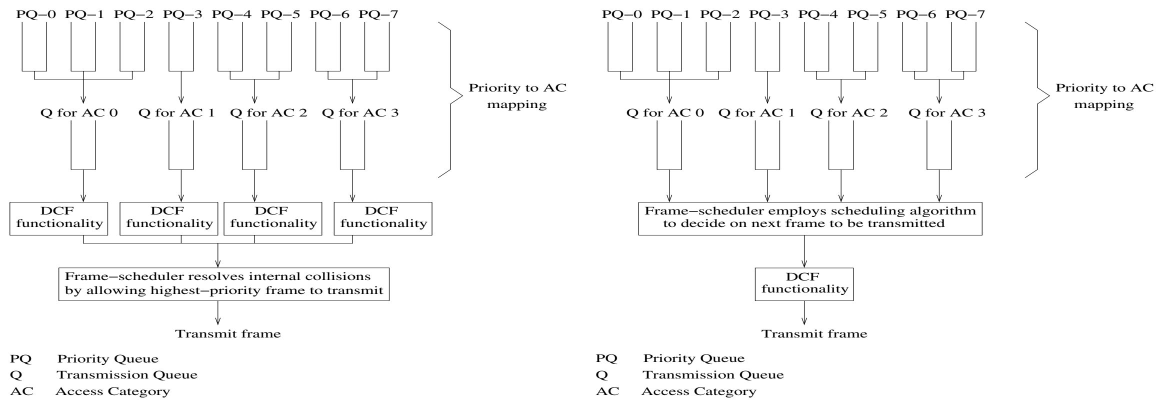

Figure 8.12: The SBB and SAB scheduling policies.

accepted and implemented in majority of next-generation 802.11 WLAN products, no QoS support is required at the MAC layer, since with a perfectly fair MAC, QoS control can be implemented at higher (e.g. IP) layers along with an appropriate admission and signalling scheme. However there is a subtle difference between having some QoS differentiation at the MAC layer and doing everything at, lets say the IP level. While in the first case (with virtual MACs and parallel contention, called `SAB` for `Schedule After Backoff`) a packet from a higher priority class can preempt a packet from a lower priority class even at the moment after the first has

[32]**In terminology of 802.11e that means the contention period of the high priority class ends before the contention period of the low priority class starts.**

already started wireless channel contention, in the second scheme (`SBB` for `Schedule Before Backoff`), class preemption happens only once, that is to say at the time the MAC signals idle channel. Therefore once a packet has been feed to the MAC by an upper-layer scheduler, nothing can be done to the in-the-station packet ordering anymore. The preemption between classes from potentially different physical MACs by means of distinct AIFS intervals must also be taken into consideration. Therefore MAC level differentiation can be advantageous for providing better real-time and voice-over-IP support. The two scheduling alternatives have been studied in the paper [85].

So what can be done to provide on the one side the behaviour like in Idle Sense with channel congestion control and on the other side, provide something like 802.11e, that is to integrate the desirable properties of both algorithms into a single solution? I think that the right answer can be found in the probability distribution approach and think that the generic approach to an optimal wireless access algorithm with configurable QoS support is as follows.

An AIMD-controlled slotted contention window scheme without the cycle-by-cycle residual backoff should be provided in each of the wireless stations in order to keep channel collisions low. Each node has the freedom to support a number of different traffic classes, which are implemented on top of the congestion controlled slotted window dynamic in size. Each traffic class can be expressed by a probability distribution function applied to the actual contention window. That is, if a packet of class 1 is passed to the MAC, its probability function $p_1(i)$ is used to draw a random slot in the actual window, while for another class 2, $p_2(i)$ is used. Once a packet is passed to the MAC layer, the MAC performs the real wireless channel contention, but if another packet arrives at the same time (this should for obvious reasons be only permitted for packets from a different QoS class), it contends internally with the first one. Also, a mechanism is necessary to resolve potential collisions between the QoS classes in one station[33] and two different possibilities are imaginable: (a) the reselection of slots and (b) the delaying of packets with lower priority (but in the second case a notion of priority distinct from the probability function has to be provided; in a stochastic approach this should be avoided). If slots are reselected, the slotted time already spent in waiting before the virtual collision has happened, must be taken into account (virtual residual backoff)[34]. Additionally short-term fairness control (or just statistical fairness) may be performed by suitable correlation management. However, it is not clear yet, how short-term fairness can be quantified in the multiple traffic classes scenario and this very likely requires a modification to the classical Jain's fairness index to include relative packet rates.

More research must be spent on investigating the above idea in practice and defining concrete relations between probability distributions and delivered QoS levels. However I think that this idea generalises efforts done in 802.11e and in Idle Sense to form a single framework. It is important to note that the stochastic approach separates two things: on the one side contention window control necessary to keep channel collisions low and on the other side, different QoS profiles providing class

[33]**If the two contending packets fall into the same slot.**
[34]**E.g. subtracted from the new backoff slot value.**

differentiation and fairness control. Further work on this subject must also include time-dependent probability distributions with suitable correlation control, which in my opinion may permit to find an improved (optimal) residual backoff algorithm based solely on probabilities.

Chapter 9

Improving End-to-End Performance of 802.11 WLANs

9.1 Motivation and Problem Description

This chapter is an attempt to deliver a QoS solution for an integrated wireless Internet gateway based on consumer 802.11 devices and to improve the perceived (by the end user) WLAN performance. An algorithm for scheduling of multiple TCP upload and download flows on an IEEE 802.11b wireless network called VFQ, which incorporates the observations about 802.11 WLAN behaviour as made in previous chapters is presented. The algorithm is based on heuristic information inference for upload flows (that is, flows from wireless stations to the outside world) by observing the wireless traffic in the download direction at the access point.

The VFQ flow regulation utilises the self clocking property of TCP connections. A TCP source will emit new segments[1] only after reception of an ACK, which indicates that an outstanding data packet has left the network and arrived at the receiver. Second key idea of the algorithm is based on the equalisation property of a WFQ-like algorithm applied to the wireless channel transmission time.

While the problem of MAC level fairness (short- as well as long-term) between nodes competing for bandwidth on a 802.11-like network has been thoroughly studied in numerous publications ([59],[59] just to name a few), the problem of the 802.11 access asymmetry between the access point and N concurrent wireless nodes has not. Recent observations as well as simulations of such scenarios reveal unsolved performance issues, if simultaneous TCP uploads and downloads have to cross a 802.11 cell.

Such utilisation scenario is quite common for an Internet access gateway, where users may have ongoing downloads, while participating in a video conference or using another application which generate TCP upload traffic at the same time. TCP is still the dominant source of traffic in an average wireless access network [12], despite the

[1] **In the remaining part of the chapter a "segment" means an outstanding, unique (defined by TCP sequence numbers) fragment of data from the sender congestion window, while a "packet" means a network datagram carrying a segment (that is an instance of) or an TCP acknowledgement. Note that a TCP packet can also carry a segment and an ACK at once.**

growing number of media applications using other protocols like RTP [117][2].

While TCP has been designed for reliable communication over an unreliable packet-based Internet and has a number of desirable properties (like theoretical throughput fairness among sessions sharing a common path, congestion detection and avoidance), it fails to address some of the specific wireless network issues.

9.2 Virtual Flow Queueing for TCP

9.2.1 Generalised WFQ Queueing

To proceed with the design of an active queue management algorithm for wireless networks, I start with a look at a generalisation of packet fair queueing algorithms. There was an active research going on in the area of fair queueing, mostly devoted to the IP and packet level. The most prominent algorithms in the area of fair queueing are the WFQ-algorithm family ([20], [49], [48]) and the probabilistic fair queueing (PRFQ, see [7], [1]) family. While PRFQ-like algorithms use a `credit model` slightly similar to a token bucket[3] to provide long-term fairness among a set of traffic classes, WFQ-like algorithms use a `debit model` to account for the past-to-now ressource utilisation of a particular traffic class. A PRFQ-like solution is computationally inexpensive (in presence of a cheap pseudo-random number generator), while WFQ algorithms require some more computation time and usually require floating point arithmetic.

The basic idea of the WFQ algorithm is the equalisation of the so called virtual time among queues served by the WFQ scheduler. The WFQ scheduler maintains a per queue weighted virtual service amount (represented by queue's q_i virtual start and finish times v^i_{start} and v^i_{end} and the queue's weight w^i) already given to that queue and always serves as next the queue, which has so far obtained the least amount of the weighted service. A global virtual time variable V is maintained to track the total amount of service given to all queues. The min-servicing property of a WFQ scheduler assures, that on average all its classes receive the same weighted amount of service. The numerous WFQ-type algorithms differ in the manner, how the next-to-send packet is selected (either by packet's smallest start or finish time) and how the per-class virtual times are updated.

The idea of this section is not to give a new WFQ-like algorithm but to generalise an existing WFQ algorithm for universal application. Since the WF^2Q algorithm and its low-complexity version called WF^2Q+ ([19])[4] has been proven to be the best packet approximation of the general processor sharing (GPS), it is used as starting point. A GPS server [104] is an idealised scheduler model, which can work with an arbitrary granularity, that is can serve any of its flows with an arbitrary speed regardless of packet boundaries (fluid model). Under GPS, if the amount of service given to a backlogged flow i ($i = 1 \dots N$) during the time interval $[t_1, t_2]$ is denoted

[2]To my knowledge used by e.g. the RealPlayer [116] in live streaming mode.

[3]Crediting is for example also used in the Linux process scheduler.

[4]In WF^2Q virtual times are maintained on per-packet basis, while WF^2Q+ maintains only per-queue times.

with $\vartheta_i(t_1, t_2)$, then the following relation holds:

$$\frac{\vartheta_i(t_1,t_2)}{\vartheta_j(t_1,t_2)} \geq \frac{w_i}{w_j}, \forall j$$

and the equality holds, if flow j is also backlogged during the interval $[t_1, t_2]$.

WF^2Q has been proven ([20]) to approximate GPS service with an accuracy bounded by the maximum size of a packet with respect to any particular flow, that is to lag behind the theoretical GPS server by not more than one maximum sized packet per class. A WF^2Q+ scheduler works as follows:

- if a new packet arrives, it is classified (by IP, port number, MAC, etc.) and assigned to one of the scheduler queues q_j (`enqueue()`)
- then the queue q_j is checked and if it is not empty, the packet is just enqueued
- if the queue q_j is empty, the packet is enqueued but a virtual time update procedure is called just after
- on `dequeue()`, the non empty queue q_j with the minimum finish time[5] is selected, the packet dequeued and the queue examined if it contains any further packets. If more packets are present, the next packet is examined and queue's virtual times updated as:

  $$v_{start}^{new} = v_{end}^{old}, v_{end}^{new} = v_{start}^{new} + \frac{L}{w_i}$$

  where L is the length of the dequeued packet and then the global virtual time is updated;
- the global virtual time is updated according to:

  $$V = max(V + \frac{L \cdot w^i}{\sum_j w^j}, v_{start}^i)$$

Since a classicl WFQ like scheduler uses the packet length to track the service already given to a particular class, it fails to provide real fairness, if the packet length does not directly represent physical transmission time, that is for example in the case of a high-overhead network link. Another example could be an ATM type link, where packets may only be of fixed size or be multiple of a minimum byte quantum, that is for example transmission of a 32 byte packet may take the same time as the transmission of 64 bytes, because one MPDU is consumed in each case. Here a naive WFQ server unaware about the physical packet transmission details would errornously provide twice the service to a flow with 32 byte packets with respect to a flow composed of 64 byte packets!

In the above situation a correction must be introduced into the WFQ algorithm to

[5]**in WF^2Q+ only queues which would have started to receive service in the corresponding fluid GPS system are eligible, that is only queues satisfying $v_{start} \leq V$.**

account for the real cost of differently sized packets. I call this type of algorithm `generalised WFQ`, if a special cost function

$$f := L \rightarrow f(L)$$

is used to calculate the virtual service amount given to a particular WFQ class. The cost function should be a positive function of the physical packet length[6] and correspond to some `observable` (a measurable quantity, channel time, packet rate, etc.) related to physical packet transmission going to receive fair service.

In the special case for the choice of $f(L) = L$ the original WFQ algorithm is reobtained. On the other hand, if $f(L)$ represents any other observable related to packet transmission, the service fairness of the underlying WFQ algorithm will also provide fairness (that is equalise the utilisation on the long-term time scale) with respect to that `observable`. Another straight example for the universal applicability of the generalised WFQ is an alternative implementation of the weighted round robin (WRR [37]): it can be obtained, if the cost function is set to the weight of the class being independent of the physical packet size, that is if: $f(L) = w_i,\ i = 1 \dots N$.

Another obvious application of the cost function approach is probably the transmission time fairness, if $f(L)$ is chosen to represent the amount of physical channel time necessary for packet transmission. I'm going to use such type of cost function including some modifications to respect the particular IEEE 802.11 WLAN characteristics for resolving the wireless TCP asymmetry problem in a mixed upload-download scenario.

9.2.2 Application to Wireless Networks: VFQ

To address the unfairness issues in TCP described above in the context of WLAN sharing, a fair queueing algorithm at the access point is necessary. But why not use something simple like standard WFQ? The answer is straight: since a 802.11 wireless network has a completely different characteristics compared to a wired link, a WFQ-like queueing discipline is not going to work well for that type of network. In particular, the high wireless transmission overhead must be taken into account. This leads to an active queue management algorithm for TCP called VFQ.

The coarse VHF^2Q architecture has been shown in the picture 9.2 and it works as follows. An active scheduling element is installed in the wireless network interface of the AP. The scheduler is composed of three building blocks: (1) a wireless node classifier, (2) node object manager and its structures and (3) an attached conventional WFQ packet scheduler. Since the VF^2Q module is interested in the TCP traffic only, it should be a sub-scheduler of a bigger compound packet scheduler installed at the access point (my Linux implementation benefits from the flexible architecture of the traffic control API, while NS implementation includes a simple priority scheduler for traffic other than TCP).

[6]A function taking negative values could be permitted in theory, however it would require more sophisticated modifications to the WF^2Q+ algorithm to incorporate non-monotonic virtual times. It may sound strange that a cost function could be negative, but for a complex protocol it could be necessary to make assumptions about the real cost of a packet therefore overestimating packet's cost while exact knowledge about the cost could be then obtained later.

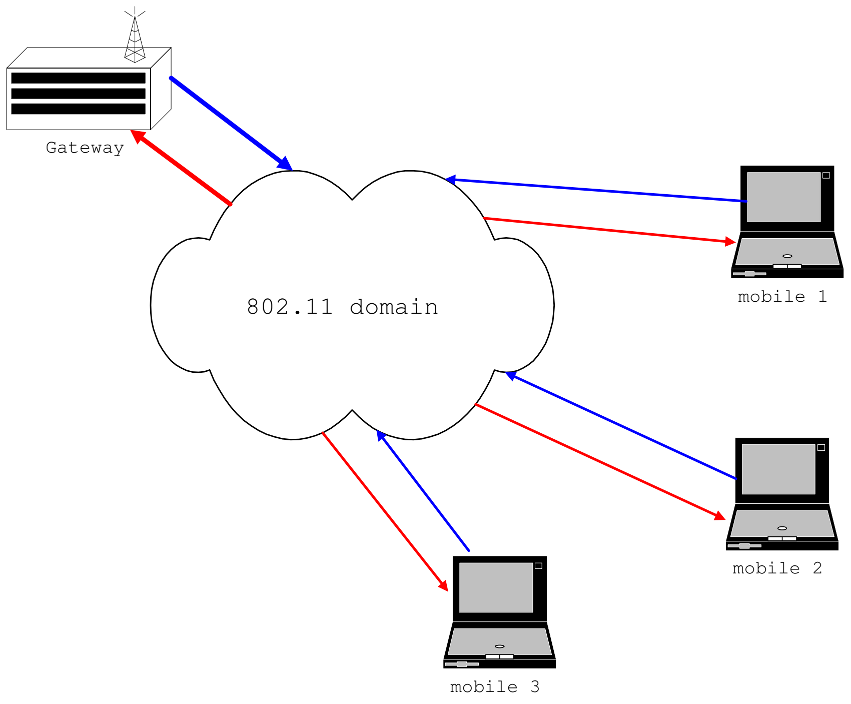

Figure 9.1: Example 802.11 network used as Internet or company access gateway for mobile nodes.

The node classifier distinguishes wireless nodes by their unique MAC address and forwards a newly arrived packet to one of the internal node objects. If for a given destination MAC no node object can be found, a new one is created on the fly (taking a configuration table into account, which stores a default node weigh set to 1 for obtaining per node wireless bandwidth fairness, however a per MAC weight setting is possible, if bandwidth differentiation between nodes is desired).

Each of the node objects is a packet scheduler too, that is provides `enqueue()` and `dequeue()` methods to the outside world. On its output side every node object (called node `black box` from now on) exposes its next outstanding packet for the attached conventional WFQ scheduler, together with an estimated packet cost η used as scheduling observable instead of the physical packet length (like in a simple WFQ setup).

The η value accounts for the node's transmission time on the wireless network and is estimated as follows:

- if the next packet available for sending and exposed by the node object is a TCP data packet of n bytes (at IP level), the η value is set to the time (most comfortabely measured in 802.11 time slots) needed to send the packet at the 802.11 MAC layer, plus the time needed (again at the MAC level) by the destination node to send the corresponding TCP acknowledgement (a standard TCP ACK packet which will be send is heuristically assumed), that is:

$$\eta = T_{802.11}(n) + T_{802.11}(ACK) \tag{9.1}$$

taking into account the following relation for the physical transmission time

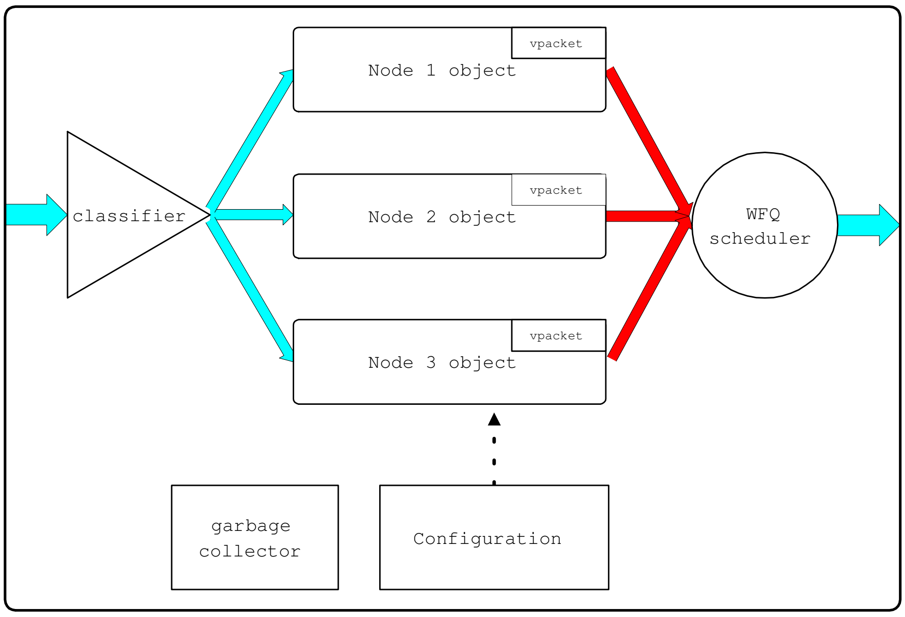

Figure 9.2: The coarse architecture of the VF^2Q scheduler.

(which does not account for MAC-level retransmissions and fragmentation so far, more on this later):

$$T_{802.11}(n) = contention(N) + phy\_time(n) + overhead \tag{9.2}$$

where the exact contention time for a particular packet is unknown, but its average value can be estimated from the actual number of active nodes N (in fact we can consider the contention wait time to follow a **2,3,4,many** rule, that is not to change too much for more than roughly 4 to 5 nodes). The overhead term in the equation accounts for the inevitable 802.11 MAC packet overhead, that is, the time needed to send a 0-bytes sized IP packet. To obtain an estimation for the MAC contention time, a simulation of the 802.11b CSMA access method was performed for an increasing number of nodes and the resulting average contention time before packet transmission has been approximated to be[7]:

$$\begin{aligned} contention(N) = 31.43747 \cdot e^{-\frac{N}{0.85404}} + 5.03185 \cdot e^{-\frac{N}{5.48378}} \\ + 1.24346 \cdot e^{-\frac{N}{61.45172}} + 0.32783 \end{aligned} \tag{9.3}$$

- if the next available packet is a TCP ACK packet of n physical bytes in length (In real-world TCP it means 40 bytes, if no additional TCP and IP options are present.), which also acknowledges m bytes from the TCP sender window, the η value is set to:

$$\eta = T_{802.11}(n) + \sum_{k_i} T_{802.11}(k_i) \tag{9.4}$$

[7]**This is just a "best fit" of the simulation data, I do not claim that the shape of the packet wait curve exactly complies to an exponential decay.**

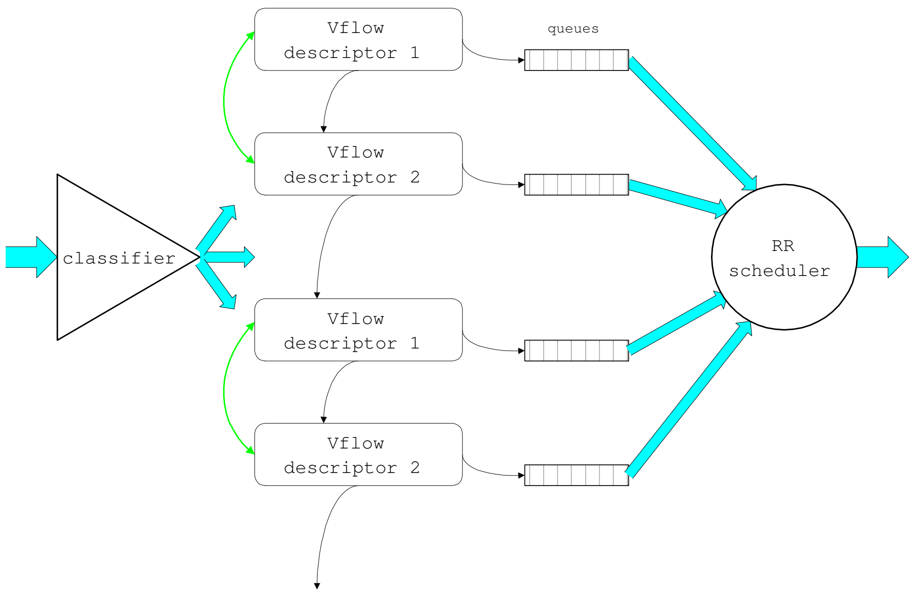

Figure 9.3: Architecture of the TCP node "black box". Two adjoint virtual flow descriptor pairs are shown.

to account for sending of the n byte sized ACK packet from the access point to the destination node and also for the m bytes of data already sent by the wireless node in the past, which permitted the ACK packet to arrive at the AP. The m bytes of data must be splitted into physical segments of some sizes k_i, but since no exact knowledge is available, how those bytes have been sent in reality, a heuristic approach is chosen, assuming that the m bytes have been sent as m/MTU maximum sized packets plus a remainder packet of $m\, mod\, MTU$ bytes.

The effective $VFQ - \eta$ value as function of the IP-level packet size and the conventional WFQ cost function (that is in fact just the IP packet size) have been shown in the figure 9.4. Since the conventional WFQ scheduler will equalise virtual times between all packet queues (black boxes) it has to serve, my approach will deliver bandwidth fairness to concurrent upload and download flows on the wireless network, if and only if the virtual queue times represent the amount of network bandwidth (or channel time, if a direct proportionality between the two quantities can be assumed) used by a particular wireless node.

Note that VFQ does not necessarily provide real channel time fairness, because it accounts only for visible, that is successfully transmitted packets thus leaving collision and retransmission overhead uncounted. Therefore the real channel time granted to a node can be written as $T' = C \cdot T$, where T stands for the accounted transmission time and C is a constant, which expresses the average overhead of collisions. If on the average all stations experience equal amount of collisions, the ratio T_i/T_j, $\forall i, j$ will still be equal to the corresponding ratio of node weights. In a 802.11 WLAN without location dependent, per-node, bandwidth and collision (or packet loss) rates, the average collision overhead will be equal for all stations. However, this picture is slightly oversimplified since in the mixed upload-download

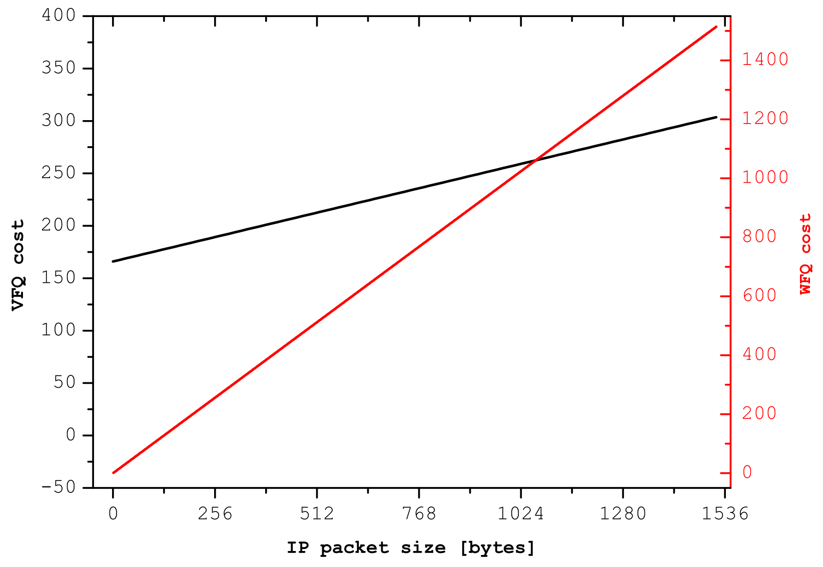

Figure 9.4: The WFQ cost function versus the VFQ cost function for a 802.11b wireless network.

case there are two types of wireless collisions:

- `long collisions` where a TCP data packet collides with another TCP data packet or an ACK packet (since under DCF the channel time lost in a collision is bounded by the duration of the longest packet)
- `short collisions` if two or more ACK-only packets collide

and it is not clear how many collisions of each type will actually happen for a given combination of upload and download flows for each station. A look at NS trace files reveals that for DropTail at the AP and N stations with $N/2$ stations downloading and the other half uploading, there are less short (ACK-ACK) collisions than long type collisions. This is logical for the observed unfair bandwidth distribution, because the biggest source of TCP ACKs are the downloading stations, which are obtaining far less of the total channel capacity.

To complete the architecture of the VFQ scheduler, the internals of the node elements ("black boxes") are shown in 9.3 and they work as follows. If a new packet for a node arrives, it is first split into an TCP ACK and TCP data part (if it is an ACK-only packet, or data-only packet, no splitting is necessary). Then for the pair of packets, a flow descriptor is retrieved from node's flow table (if no flow descriptor for a given TCP flow quadruple, that is $[IP_{src}, port_{src}, IP_{dst}, port_{dst}]$ combination exists, a new descriptor pair is created on the fly) and the packet(s) are stored into the appropriate flow queues[8].

[8]**In a real implementation partial buffer share with just one queue for all flows can be chosen and the flow objects made just virtual flow tracker objects.**

For a given TCP flow there should be always a pair of linked flow descriptors (one for TCP data packets sent by the AP onto the 802.11 channel and a second one for ACK packets traversing the AP in the other direction), because a TCP connection is in general bidirectional. However, the actual NS2 prototype supports only the unidirectional variant of NS2' TCP for the sake of simplicity[9].

The ACK flow descriptor must further track the actual sender window size in order to preserve some minimal TCP security[10] and recognise duplicate ACK segments. The issue of tracking of TCP flows by an intermediary station has already been subject to numerous papers and found to be (in theory) not perfectly solvable. However, since the considered network type is fairly low-speed, tracking of the ACK sequence numbers and the announced window size is in practice straightforward (because window scaling is usually not used and the window can grow only up to about 65536 bytes). On the other hand, simple TCP flow tracking is already state of the art in majority of wireless products implementing any firewall or NAT functionality[11], so that VFQ-type queue management can be seamlessly integrated into existing solutions.

Some additional care must be taken, if the outstanding packet is an ACK acknowledging a whole bunch of data segments. Because the scheduling should not become too coarse (that is the η value should not become too high to keep the packets for a station continously flowing), ACKs are allowed for up to a configurable amount of data packets, which was set to four MTU segments in the remaining VFQ simulations. Thus the waiting ACK segments from node's flow queues must be broken down into smaller ones, in order to prevent too rough TCP bursting. This may appear strange, but the normal point of operation of a TCP source is the reception of one (two in the case of a Del-ACK node that is delayed-ACK policy) ACK packet for each data packet sent, so that this restriction only restores the intended point of operation for TCP.

A special case may occur, if there are data and ACK packets waiting in the two adjoint flow queues at the same time (this is in fact the practical reason for having those two descriptors always linked to each other). Since in practice the wireless bandwidth is a scare ressource, it is advantageous to combine the two packets again into one physical TCP packet[12] (that is setting the ACK flag and updating the ACK sequence number field in the TCP header together with the checksum[13]) and also to accordingly update the node's η value visible to the attached WFQ scheduler, in order to account for the additional ACK cost.

Last issue to be discussed are duplicated ACKs and TCP retransmissions. If a duplicate ACK (that is an ACK with a sequence number from the bottom of the actual tracking window) arrives, some care must be taken, because duplicated ACKs will normally force the TCP sender to retransmit some more packets. In VFQ this case

[9]In NS2 the TCP modules other than Reno provide also only one-way TCP.

[10]Against segment spoofing, which requires the knowledge of the correct sequence number.

[11]In Linux mature connection tracking modules are available for the Netfilter [98] subsystem.

[12]A practical Linux implementation may deal with that case more intelligently, that is may avoid any packet splitting and reconstruction.

[13]Here the Linux kernel provides an efficient recursive checksum updating function, if just few bytes have been changed in a datagram, without the need of complete checksum recalculation.

is accounted for by assuming that a DUP-ACK will trigger retransmission of the missing segment (the amount of retransmitted packets can however be made configurable in a future version). It is important to note, that different TCP senders have mostly identical retransmission behaviour, with sole exception of Vegas-TCP, which may already retransmit before three DUP-ACKs appear - this point must be further investigated.

One more practical problem to be solved is the scheduling of flows inside a particular wireless node object, because a station may open more than just one TCP connection at once. I have chosen a simple round robin (RR) scheduler for this purpose. That is not an optimal solution for all possible scenarios, but since I'm only concerned about the fairness problem between physical wireless nodes, this issue is of less importance and left for further research.

9.3 Validation and Prototype

In this section simulation results from the NS2 VFQ prototype as well as from the Linux kernel implementation are presented and studied in more detail. The topology of the simulated network is the same as for the previously studied drop tail queueing case in the AP (fig. 7.1). The maximum, per node VFQ queue size was set to 10 packets, in order to provide equal amount of packet buffers in the access point as in the drop-tail simulations (where for N nodes the queue size was set to $10 * N$), otherwise the comparison wouldn't be fair.

9.3.1 Validation in the NS2 Simulator

NS2 Architecture of VFQ

The VFQ algorithm has been implemented in the NS2 network simulator ([93], [27]) version 2.28 using the architecture hints provided in the general VFQ description. The VFQ design easily fits the NS scheduler architecture, which requires a minimalistic scheduling module to provide at least an enqueue and a dequeue method. An integrated priority scheduler inside the VFQ class provides absolute priority to packets other than TCP (used for example for routing inside NS2[14]). A limitation exists in NS, because NS provides only a segment-level approximation of TCP, that is, in opposite to real world implementations, NS's TCP always sends fixed size chunks of data. Also the TCP congestion window size is managed in packets[15], rather than in bytes and the same limitation holds for the acknowledgement number transmitted in TCP packet headers.

The NS VFQ implementation is highly configurable and exposes following tunable parameters to the TCL script used to launch the simulation. The parameters can be set through the `Queue/VFQ set` command: The NS code of the VFQ module is mature and stable but has some limitations too: the weight setting for a station

[14]In contrast to a real IEEE 802.11 network, the NS 802.11 module can operate in an ad-hoc routing mode, where other wireless nodes forward packets to other stations not in the range of the access point node.

[15]However NS TCP uses floating point numbers for the window size, including fractions of the segment size.

Table 9.1: VFQ TCL configuration parameters in NS

variable name	default value	meaning
DownloadWeight	1.0	relative download node weight
UploadWeight	1.0	relative upload node weight
BoxMaxLen	10	max. per WLAN node queue size
QueueLength	500	global VFQ packet limit
BoxesMaxCount	-1 (unlimited)	limit of served WLAN nodes
SimpleWFQ	false	switches from VFQ back to WF^2Q+
MACOverhead	0.0	cost function base
TCPsegsize	1500	assumed TCP packet size
TCPackburst	8	max ACK burst size
BcastQueueLength	100	int. Prio queue length

during the lifetime of the VFQ class depends on the TCP flow direction, which VFQ observes at the time it must create a new node object, that is, if a node enters the simulation by opening a download TCP connection, it will have the value of `DownloadWeight` assigned, otherwise the value of `UploadWeight` is assigned to the station[16]. However, this is absolutely sufficient for the simulation of the desired upload-download scenarios with different upload and download priorities.

The VFQ implementation has been also validated by checking the debugging output from the code for two sample simulations with 3 upload and 3 download stations. First of all, the proper operation of VFQ in a pure WFQ mode was assured by tracing a sequence of enqueue and dequeue operations and checking the queue virtual times. Second, the VFQ virtual mode was enabled and the same sequence[17] of operations was checked, while taking the virtual packet sizes (that is the cost) into account.

NS Simulation Results

An initial simulation has been performed for the simple scenario nr. 1 with six download-only NewReno flows to check the correct functioning of the prototype. The preliminary finding is that under the VFQ discipline at the AP the inherent unfairness between few (New)Reno flows vanishes and that the stations obtain mostly perfectly equal bandwidth shares (this is, however the same result as if a conventional WFQ packet-level scheduler would have been used at the AP). I do not show the corresponding time-volume figure here, because it looks very close to the figure 9.5 as presented for the scenario nr. 3.

Next simulation scenario studied is the 3 download and 3 upload flows case (scenario nr. 3). As depicted in the figure 9.6 showing the evolution of the fairness index over time but also directly visible from the corresponding time-volume graph 9.5 there is an obvious improvement in the achieved short-term[18] fairness between

[16]A per MAC or IP weight setting is desirable.

[17]Same in the sense that the same PRNG seed value was used to initialise the NS2 simulation.

[18]But also long-term since the absolute amount of bytes transferred at the end of simulation equals between all the six flows.

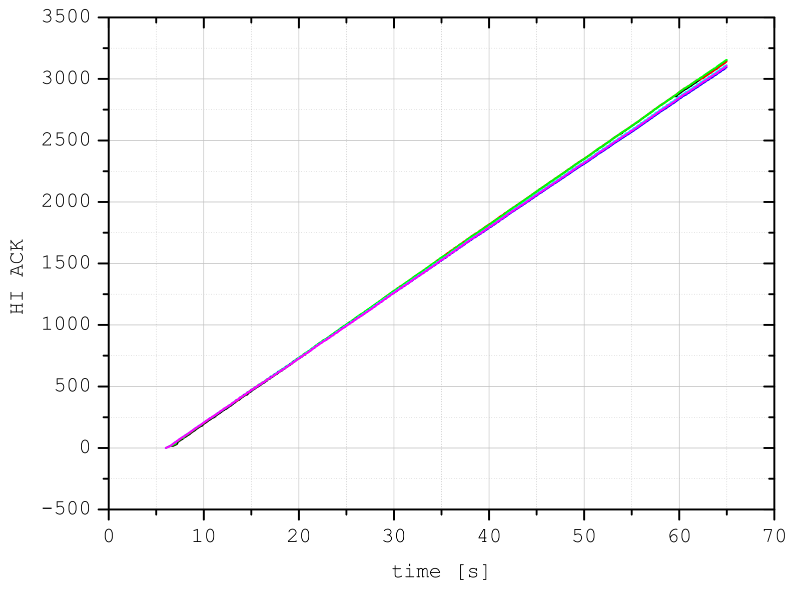

Figure 9.5: Scenario 3 under VFQ queueing discipline, 3 download and 3 upload flows obtain perfectly equal bandwidth (the 6 curves overlap).

the upload and download direction. The two types of TCP sessions are now able to receive a nearly equal bandwidth shares.

The price one has to pay for the short-term fairness in VFQ is an asymmetry in the RTT (round trip) estimates experienced by the two flow directions as can be seen in the figure 9.7. For a real-world TCP implementation the possible delay experienced by the sender is actively limited by the receiver-side flow control, which will bound the maximum number of data packets in-fly the source is permitted to emit to resemble the available buffer space in the receiver-side socket. In contrast, in a NS2 simulation like the one presented here, the delay is only limited by the setting of the `maxcwnd_` parameter in the NS2 TCP module, which statically limits the emulated receiver-side buffer space (static flow control).

A closer look at the fairness index plot in the figure 9.6 reveals a slight problem of the VFQ solution: the fairness index is not really equal to 1 at the very beginning of the simulation and starts at a value of about 0.95. However, the slight unfairness at the very beginning is not directly visible from the corresponding time-volume plot and thus simultaneously provides an insight into the high sensitivity of the Jain's fairness index approach. The observed effect can be explained as follows: since VFQ can regulate the upload flows only by their ACK packets, at the very beginning of the simulation the three uploading stations can quickly send out their initial window[19] of data and immediately profit from their equal shares of the 802.11 WLAN packet rate, before they can be actually throttled down by the VFQ server installed

[19] **It was set to 4 segments in the NS2 TCP module.**

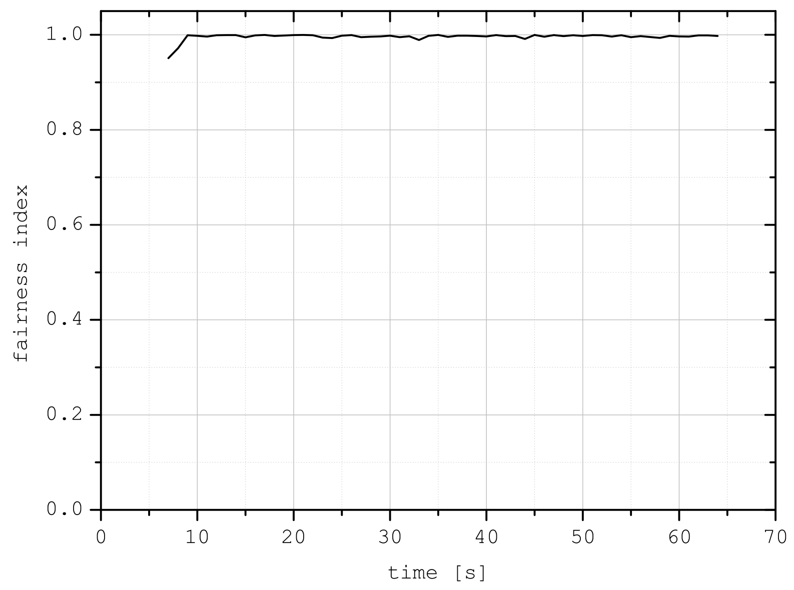

Figure 9.6: Jain's fairness index for 3 downloading and 3 uploading TCP wireless nodes under VFQ queueing.

in the AP. The effect is of course the smaller the lower the setting of the initial TCP window size, which may lie between 1 and 4 segments according to TCP RFCs. Nevertheless, the initial fairness provided by VFQ seems satisfactory.

As next it is interesting to look at the development of the sender side congestion window size over time. As shown in the figure 9.8, the VFQ service discipline in the access point tends to desynchronise the growth of the congestion window size between the participating sessions[20] and therefore provides a much more stable round trip delay compared to a pure drop tail queue setup. This can be understood as follows: since VFQ equalises the virtual channel times of all stations, it is not very probable that two packets belonging to the same wireless node will be send in sequence, that is, VFQ transforms possible bursts of TCP packets from a given station into an interleaving sequence of packets from different nodes. Therefore wireless stations are most likely served in a round-robin fashion, if they send equally sized TCP packets and exhibit same ACKing behaviour. Thus the growths of the respective congestion window sizes of the participating nodes are interleft as well.

The goal of the next simulation shown here is to confirm the theoretical ability of VFQ to provide weighted fairness between the physical stations, if desired by the user. For this purpose the simulation script used to start the scenario 3 has been modified to assign a weight of 2 to the WLAN stations performing a TCP download, while keeping the setting for upload weight to be 1. The simulation result shown in

[20]This is so far only true for one TCP flow per node since each node has its own dedicated queue at the AP.

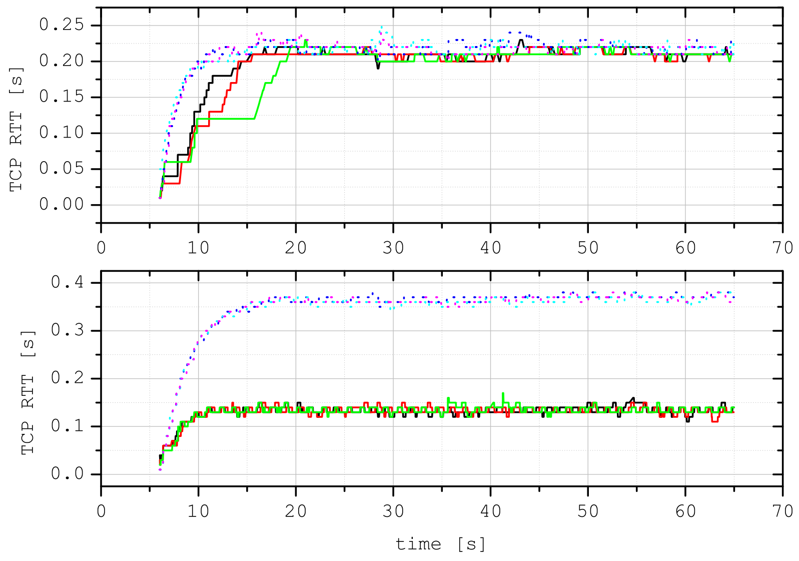

Figure 9.7: Comparison of the smoothened TCP RTT estimates for upload (dotted lines) and download direction. Upper graph: 3+3 nodes under drop tail, lower graph same scenario under VFQ.

the figure 9.9 is not so surprising, confirming that the uploading stations are gaining perfectly half of the volume obtained by the uploading stations. This is however very interesting, since in an "out-of-the-box" 802.11 wireless LAN, the resulting performance of TCP downloads and uploads is exactly the other way around.

For this reason the VFQ scheduling gives an effective knob to the wireless network administrator, which he can touch in order to limit the bandwidth used by flows in the upload direction. This is particularly interesting for limiting outgoing TCP services like peer-to-peer networks, FTP or any other server type applications, which may be running on the mobile nodes. If no such precautions are taken, even very few users of a WLAN cell can render the WLAN rather unusable, if they run just few of those upload-oriented services. This problem may appear in places, where a 802.11 WLAN is used to provide seamless Internet connectivity to a group of rather unrelated and uncoordinated participants (that is persons unaware about the needs and actions of the other network users) like in a student residence or even the exemplary airport or rail station hot spot. While WLAN technology is believed to be sufficiently mature to replace wireline access in those scenarios, this simple example shows, that rather the opposite is the case and that actual WLAN products based on IEEE 802.11 are not fully able to provide satisfactory performance without additional precautions like an active queue management at the AP.

Last scenario where the behaviour of the VFQ queueing is interesting to study is an asymmetric backbone network, where some of the TCP flows reaching the wireless domain have to cross a high-delay link while the other competing flows experience lower link delay. It is well known that (New)Reno TCP is biased towards sessions

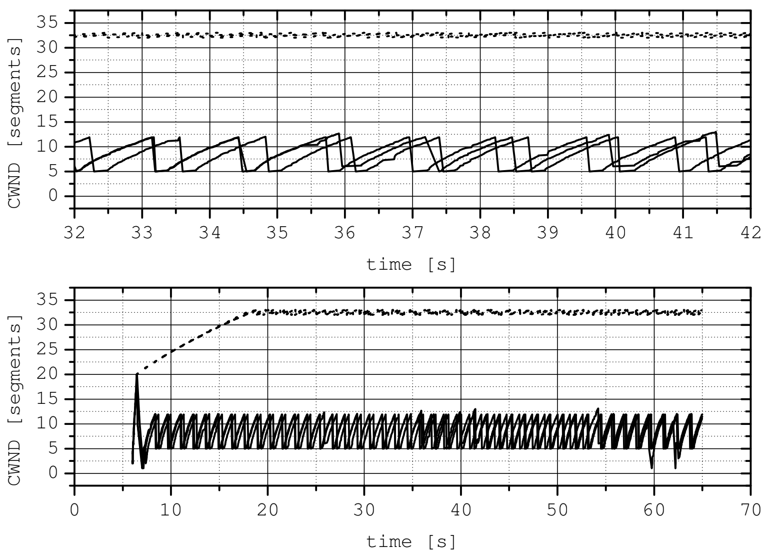

Figure 9.8: Scenario 3, congestion window development under the VFQ discipline.

with smaller RTTs [51], [78] and exhibits only a $\frac{RTT_1}{RTT_2}$ fairness property, that is the bandwidth shares of sessions crossing a bottleneck are proportional to the quotient of the total path RTTs (and as shown in the article [51] Westwood TCP exhibits only a $\sqrt{RTT}$ dependancy on path RTT under same packet loss conditions). In order to simulate this situation in NS, the topology used in the initial scenario nr. 1 has been extended to use two distinct web servers with configurable links connected to the base station through an intermediary router node[21]. The simulation script has been modified to permit the definition of which of the stations should use either the first or the second available web server performing either a download or an upload.

In order to confirm the behaviour of Reno-TCP for different RTTs as known from the literature, an initial simulation has been executed for the two link delays set to 10ms and 100ms respectively, with a capacity of 10Mbps each. As can be seen from the figure 9.10, under drop tail queueing at the access point the performance of Reno-TCP significantly suffers from the RTT asymmetry, however, the strong RTT_1/RTT_2 dependancy known from the literature cannot be fully confirmed in this simple NS simulation.

This effect can be explained as follows: the base link delay as specified in the simulation script, is not the delay effectively seen by the TCP sender, but only contributes to the total delay experienced by the source, because of the already mentioned `self-congestion` phenomenon. While in an ordinary Internet scenario a TCP session is most likely to traverse several routers and bottlenecks along its path and can, on the average, contribute just very few packets to the overall number of packets in transit at a given link, in a wireless scenario where the network path is not congested

[21] **The attempt to avoid the use of an additional intermediary router node failed due to the limitation of the NS 802.11 module permitting the base station to have only one wired link.**

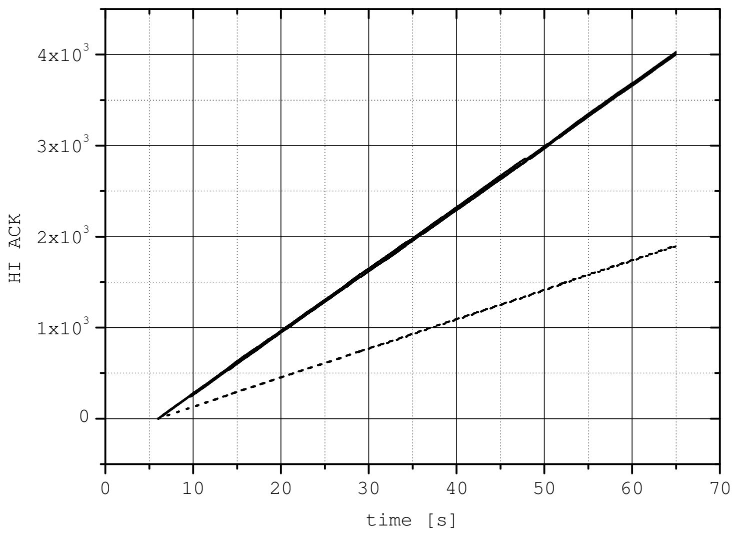

Figure 9.9: Scenario 3, VFQ with weights set to 2 for the down- and to 1 for the upload direction.

at all before the TCP source starts to emit its packets, a single TCP source induces its own congestion (that is queueing delay), which seems to increase the visible path RTT and is measured later as the effective network delay. Therefore a few TCP sessions sharing a previously non-congested network but experiencing different base link delays exhibit better long-term throughput fairness than it could be expected in a many-flow large scale Internet.

This picture changes dramatically, if the VFQ scheduler is installed in place of the drop-tail queue at the AP: indeed, the VFQ scheduler is able to compensate for the RTT asymmetry effect in the case of few Reno sessions as depicted in the figure 9.12, where the base link delays have been set to 10 ms and 50 ms respectively. However, the VFQ regulation does not work perfectly, if the delay difference between the sessions becomes too large. As can be observed in the figure 9.11 where the base link delays have been set to 10 ms and 100 ms, the VFQ server starts to fail to establish perfect long-term throughput fairness. This can be explained by the assumption, that some of the VFQ's queues are not always backlogged during the duration of the simulation run. Therefore the faster sessions (the sessions with the lower base link RTT) can receive additional bandwidth as soon as some of the internal VFQ queues become inactive and the integrated WFQ scheduler starts to distribute excess bandwidth among the active ones.

9.3.2 Linux Experimentation

Experimental Setup Description

After a simulative validation of the basic idea for a weighted fair TCP scheduler, two experiments have been carried out with real-world equipment to investigate the performance of the VFQ algorithm in a 802.11b network environment. The

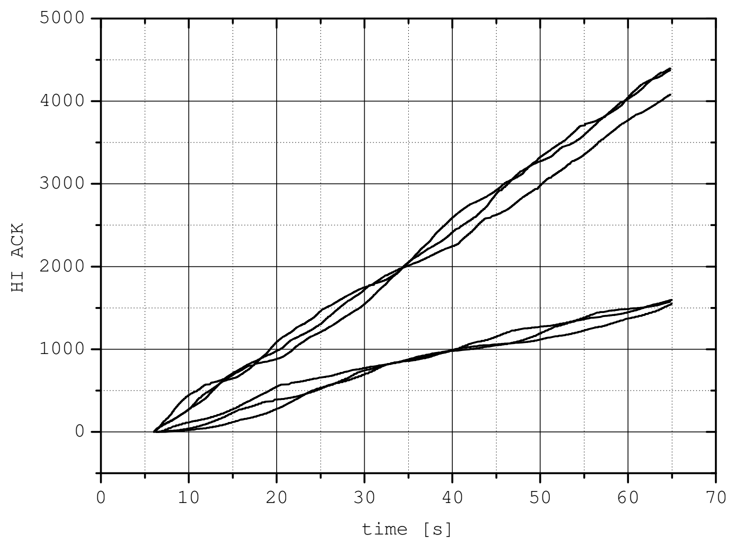

Figure 9.10: Simulation with two distinct web servers: drop tail at the AP, 3 Newreno flows with link delay 10ms, 3 flows with 100ms

basic problem of the experimentation was the availability of a significant number of portable machines and the practical organisation of the experiments.

The role of the base station played again the P2 machine equipped with the Prism2 based Compaq WL100 card, while for the mobile nodes a set of Dell Latitude D600 portables with integrated Centrino cards and SuSE Linux was used. However, to provide a more realistic scenario, two of the mobiles were equipped with different cards: one with a Cisco A350 PCMCIA card and one with a Compaq WL110 card, in order to better approach a hot-spot wireless cell, where a more or less random mix of different hardware devices can be expected in practice. In contrast to the previous 802.11 basic evaluation part, the laptop machines used the Linux kernel version 2.6.9 already incorporating the TCP Westwood extensions. Unfortunately, only 4 portable machines were available for experimentation during the time period reserved for the practical experimentation, so that a more comprehensive test of VFQ was not possible. However, even with this spare equipment I was able to provide initial validation of VFQ algorithm on a real 802.11 equipment.

For the software part at the base station side an Apache server version 1.3.31 was installed to provide a simple TCP-download data source over HTTP as well as a TCP sink (for TCP uploads) implemented through a single `NULL CGI`, which is a few-line CGI [34] script just discarding the incoming data stream. Since the base station was still running the kernel version 2.4, this experiment provides also an opportunity to test the behaviour of different versions of TCP in a mixed download-upload scenario. The kernel version at the base station did not include any Westwood-TCP extensions. The experimental data was gathered using the HTBench software described previously. The four laptop machines were synchronised with a broadcast packet sent over the WLAN.

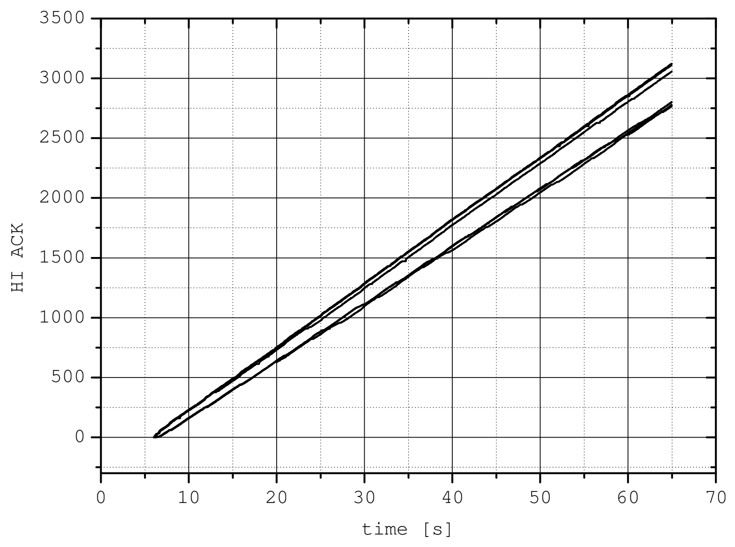

Figure 9.11: Two distinct web servers, VFQ, 3x Newreno at 10ms, 3x at 100ms

Linux VFQ Architecture and Implementation

The Linux version of the VFQ scheduler has been designed and implemented as a two-component queueing discipline architecture based on the Linux Traffic Control framework ([65]) and consists of a dynamic WF^2Q scheduler module ("VFQ shell"), which provides per-destination address TCP packet classification as well as the WF^2Q scheduling algorithm based on a cost function value provided by the second module called VPSE for `Virtual Packet Size Estimator` implementing the node scheduler object ("black box"). While actually I consider myself to dispose over some general experience in coding at the Linux kernel level, the practical implementation and debugging of the VFQ kernel module was still a hard task.

The WF^2Q part of the TCP scheduler exposes a classful queueing discipline (see [65] for more details on this issue) to the Linux kernel and provides a simple packet classifier, which classifies the incoming TCP packets according to their destination IP[22]. The internal classes of VFQ representing the wireless destination nodes and containing instances of the node scheduler object (VPSE) are created on the fly, if a new destination address is detected in the `enqueue()` method of the VFQ module. A simple garbage collector is included in the code to clean up (remove) node objects after a definable period of inactivity (that is a period of time where no packets for a given destination arrived). The VFQ scheduler "shell" dynamically adapts to the changing number of internal node objects by adjusting the virtual time update procedure with respect to the number of active nodes. This differs from other fair-queueing modules actually available for the Linux kernel, which usually require that the number of classes going to receive fair service is fixed and defined at the moment

[22]**This also assumes that wireless nodes have just one IP address, but a modification to use the MAC address instead is straightforward.**

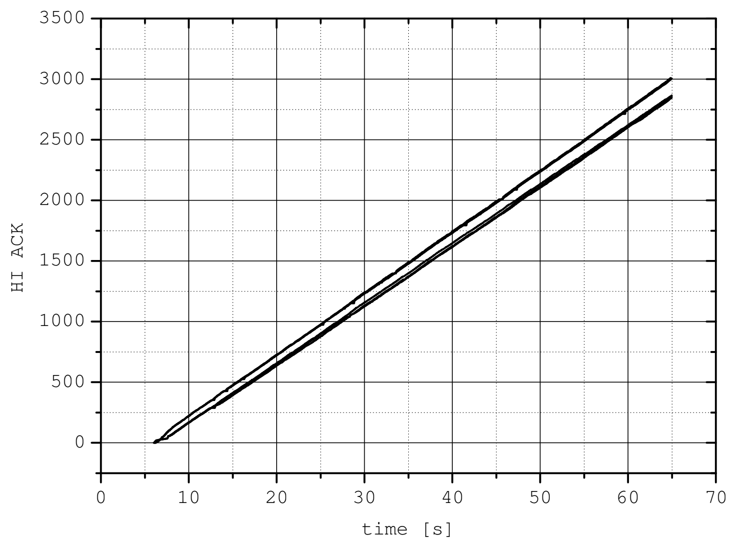

Figure 9.12: Two distinct web servers, VFQ, 3x Newreno at 10ms, 3x at 50ms

of either the module loading or the installation (instantiation) of the module in a network interface (by issuing appropriate `tc` commands).

The node objects ("black boxes") are implemented by instances of the VPSE queueing module and the outside WF^2Q "shell" must be told about the type and the existence of the right node object to use by creating an initial instance of the VPSE object for a special class (with the reserved classid of 0) inside the WF^2Q "shell". The initial instance of VPSE is then used as a reference[23] for creating more node schedulers inside the WF^2Q object, if new wireless stations are detected. However if desired, any other Linux queueing discipline can be used for generating node objects inside the VFQ server if provided as the reference object. This architectural choice permits easier experimentation and validation of the Linux VFQ module, while being easily extensible to for example provide a different cost function for use on a different type of wireless network. For this purpose the user may have for example loaded two different types of the VPSE module into the kernel, e.g. one for IEEE 802.11 and one for Bluetooth and then on the fly decide, which one is provided as reference to the VFQ object for a given network interface.

The communication between the enclosing VFQ shell and VPSE objects is accomplished by the `tc_index` field included in each Linux socket buffer (`skb`), which is a placeholder for network packets at the kernel level. The `tc_index` field is normally used for the communication between different queueing modules and is a general-purpose 32bit field available in the `skb` structure (present only if the kernel has been

[23]**This is related to the internal architecture of the Linux kernel network layer which requires that a reference to an "operation structure" ("ops") is provided when creating a scheduler object. The operation structure basically provides methods like `enqueue()` and `dequeue()` but also methods to correctly initialise the object. The type of the "ops" decides, which scheduler object among all the registered schedulers is finally created and returned to the caller.**

compiled with traffic control enabled), however it has also found an application in the routing code of Linux in cooperation with the Netfilter code ([98]), where it is used to carry routing cost information.

The VPSE node object always updates the `tc_index` field of a `skb` if asked by an upper-level module to release a packet (via a `dequeue()` or a `peek()` operation). The VFQ "shell" stores the number of active wireless nodes in the `tc_index` field of each `skb` on `enqueue()` of a TCP packet into one of the node objects. Since the cost of a packet calculated by VPSE is never below some minimum value (sending even a hypothetical zero byte sized packet at the IP level causes a substantial wireless overhead)[24] and the permitted number of wireless stations is limited to some maximum value (defined at the compilation time), the VFQ shell can distinguish the value returned in the `tc_index` field after a node object releases a packet to be either the initial number of active nodes stored there on `enqueue()`, or to be the packet cost calculated by the VPSE object. The number of active wireless nodes must be provided to the VPSE object, in order to properly estimate the time spent in the DCF contention (average number of idle slots). By using these intelligent min-max `tc_index` bounds, the VFQ "shell" can easily cooperate with node modules other than the VPSE not returning any virtual packet cost, reverting it to just a conventional WF^2Q scheduler honoring just the physical packet length.

Measurements

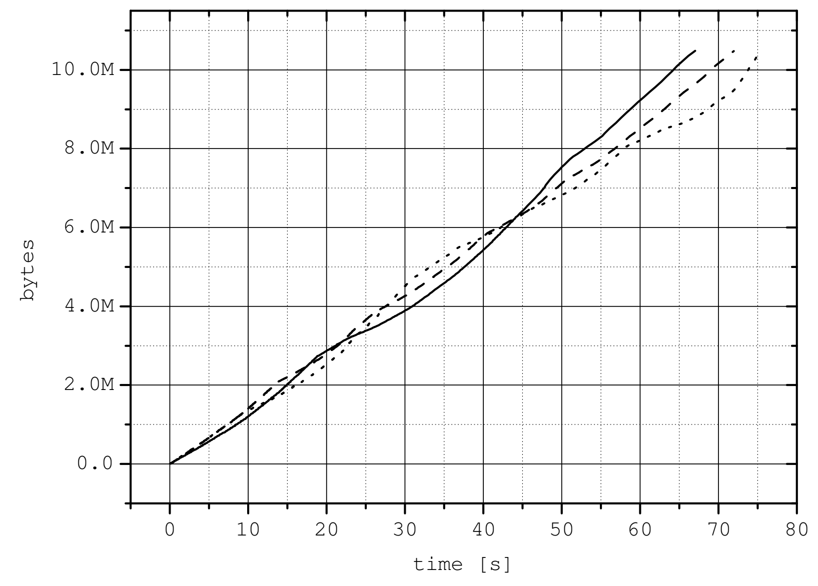

Figure 9.13: Linux TCP transfers over 802.11b, Drop Tail at the AP, queue length set to 30 packets.

[24]Actually the cost calculations are performed using integer arithmetic with an appropriate scaling factor which yields a packet cost expressed in microseconds.

The goal of the initial experiments under real world conditions was to confirm the upload-download asymmetry already observed in NS simulations as well as to acknowledge the existence of a short-term fairness problem of Reno-like TCP in the presence of few concurrent flows. The resulting figure 9.13 (for two Centrino and one Compaq card) confirms the inherent unfairness between multiple but very few Reno connections. The variation in the experienced download bandwidth over time between concurrent TCP flows are of similar order of magnitude as in previous NS simulations, however its exact strength depends on the particular choice of the wireless cards. Most probable explanation to this phenomenon is a difference in the implementation of the 802.11 standard by various hardware manufacturers, especially problems with cards' PRNGs. While I was not able to derive an exact prediction about the TCP-level performance of a WLAN card based on its DCF fingerprint, it can be expected that a faulty implementation (with respect to weak PRNG implementation[25]) will provide lower throughput than a product fully conformant with the IEEE 802.11 standard. Previous throughput measurements (table 6.1) already suggested that performance differences between the cards at the packet level (like an UDP throughput test) have an impact on the TCP throughput too.

In the second experiment the upload-download asymmetry like observed in NS sim-

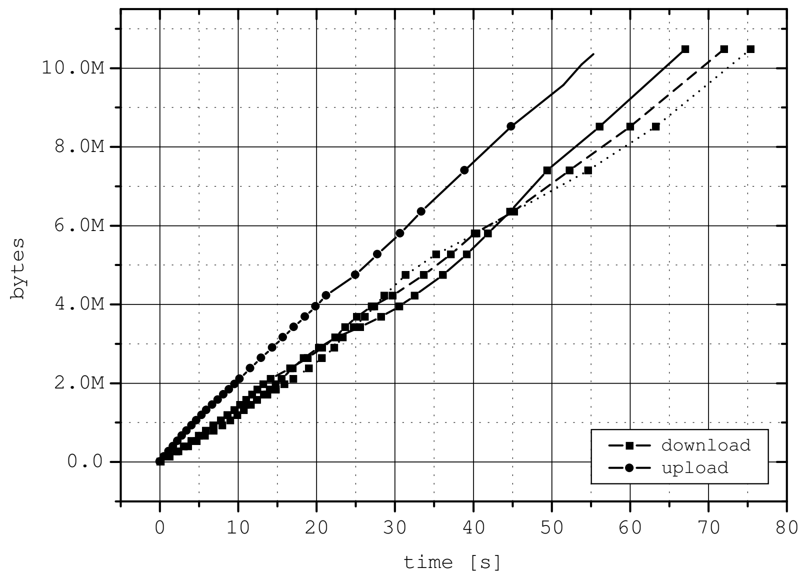

Figure 9.14: Linux TCP transfers over 802.11b, 3 downloads and one upload, DropTail q=40 packets.

ulations could be confirmed under real conditions and the resulting volume diagram shown in the figure 9.14. While having only four mobiles it was logical to either study the case of one uploading and three downloading stations or the opposite, to

[25]**of course it is possible to incorrectly implement IEEE 802.11 and deliver increased performance, e.g. by shortening the IFS spaces, however such equipment may perform poorly in a mix with IEEE conformant devices.**

obtain the most asymmetric situation. For only four machines the observable effect is not so strong as in preceeding NS simulations with 6 wireless stations, however it is also clearly visible and significant. The TCP upload volume curve appears to be much more stable (more continuous) than for the download direction and this can be contributed to the lower sensitivity of the upload flow to packet drops at the access point queue. The per-station download volumes seem to slightly stabilise over time but this is more an artifact of the HtBench TCP client than a real effect, since HtBench uses a higher sampling precision at the beginning of the TCP connection. This feature has been included in HtBench to provide higher sensitivity during the assumed slow start phase of a TCP connection. It may be curious to see that the download curves do not change their slope, once the upload connection is apparently finished with the data transfer, but this effect is again coming from the behaviour of HtBench, which will establish another upload connection (which is discarded if it cannot be finished until the end of the measurement) once the previous transfer has completed (otherwise the experimental conditions would change in the middle of the measurement).

The third experiment which I have conducted provides a practical test for the hy-

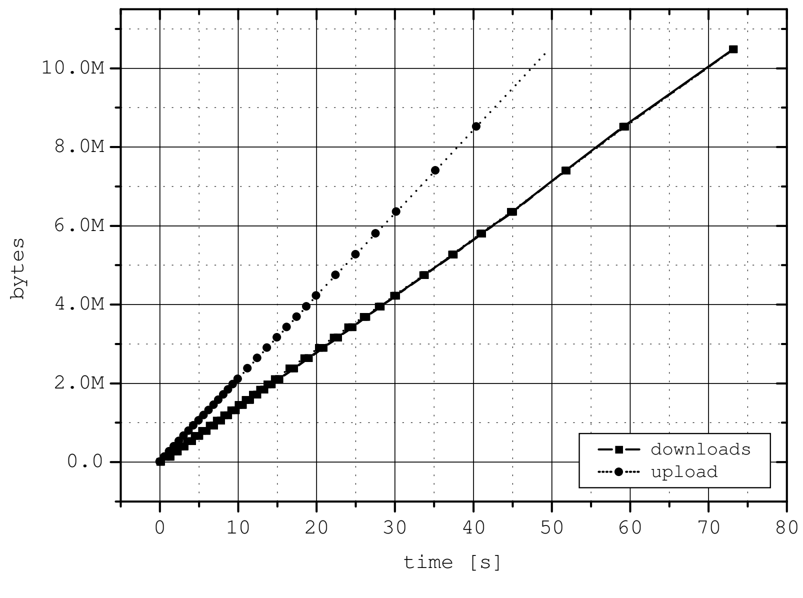

Figure 9.15: Linux TCP transfers over 802.11b, 3 downloads (curves overlapping) and one upload, simple WFQ, queue length=10 pkt. per class.

pothesis, that a simple WFQ scheduler cannot solve the upload-download asymmetry problem in an infrastructure IEEE 802.11 network. The corresponding time-volume plot in the figure 9.15 reveals even a higher difference between the downloading and the uploading connection's times (note that both connection types transfer exactly 10 MByte of data) than without any active scheduling at the access point, while the short-term unfairness among the three download connections has been eliminated (therefore the three download curves perfectly overlap). The huge gap between the two total transfer times (and therefore between the obtained bandwidth shares) of the download and the upload connection can be again contributed to the use of the physical packet length in the simple WFQ server to account for the respective virtual service times.

Finally, the last experiment for the same combination of upload and download flows

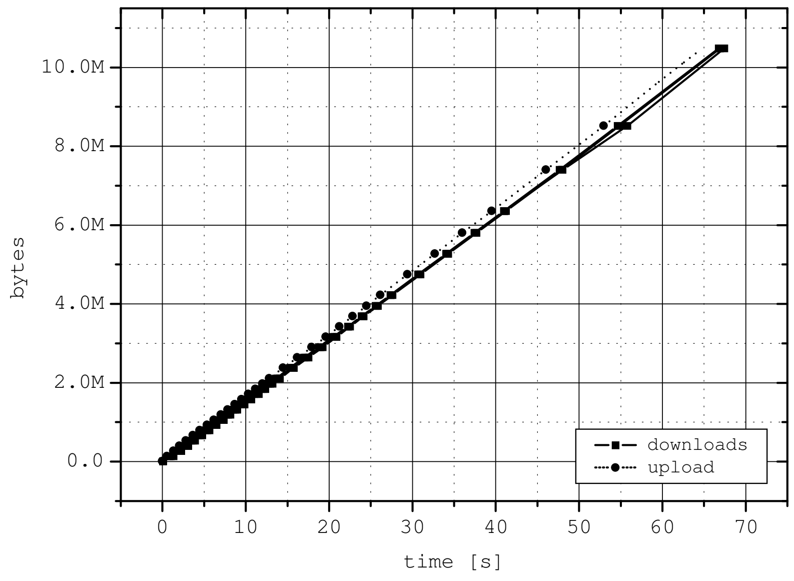

Figure 9.16: Linux TCP transfers over 802.11b, 3 downloads and one upload, VFQ with queue length=10 pkt. per station.

was carried out with the full VFQ module loaded and inserted into the packet path of the wireless interface of the base station with queue sizes set to 10 packets per unique wireless destination. The result is depicted in the figure 9.16 and confirms the simulation data obtained with NS, which have already proven that the VFQ service discipline can provide fair but also weighted TCP throughput allocation to wireless stations in a mixed upload-download scenario.

9.4 Other Solutions for the TCP Asymmetry Problem

Another two ad-hoc solutions for the upload-download fairness problem in TCP have been proposed in the publications [110] and [38]. The first article proposes a lossless active queue management scheme exploiting the announced TCP window size[26] information carried by ACK packets in the upload direction. By dynamically modifying the window-size field in the TCP header a centralised queue manager in the AP is able to constraint the effective bandwidth of TCP flows in the upload direction, therefore leaving more transmission time for itself and the download di-

[26]The amount of data that the TCP receiver can accept regardless of the size of the sender-side congestion window.

rection.

While this approach seems to work according to the result presented in the first article, it is more complex than the VFQ solution presented here and requires frequent packet manipulation and thus frequent header checksum field updates. In contrary, the VFQ service discipline requires packet manipulation only in case of heavy bursts of data packets but can even work without performing any packet manipulation, if a more coarse scheduling can be accepted by the user. It is not clear, whether the control approach by modifying the announced TCP window size may interact badly with some types of applications. A big disadvantage of the lossless approach is the inability to account for location dependent channel capacity (e.g. one station sending at 2Mbit but others at 11Mbit) and also to easily provide a priority scheme like VFQ does. Moreover, the solution [110] does not really guarantee fairness, if the stations use differently sized TCP segments (e.g. due to different path MTUs or the application level setting).

The second solution [38] discussed here goes into similar direction as VFQ does by deploying an IP-level rate limiter at the output of the AP wireless interface, in order to control the bandwidth of the ACK and the data packet streams separately. The design goal of the solution 2 is to distribute the available bandwidth among flows, not nodes, but in a 802.11 like wireless network the number of physical nodes has a very huge impact on the achievable performance due to the DCF contention overhead and must be accounted for. Moreover, the article studies only the case, where the amount of download and upload flows is equal[27]. Each of the two possible data paths (ACK+DATA) in the AP has one dedicated FIFO queue of a fixed length and may therefore suffer from Reno-like unfairness, if the queue size is chosen too small[28].

In contrary, the VFQ scheduler uses one queue object per wireless destination (MAC) and this is the right policy for an IEEE 802.11 WLAN, since each wireless destination includes also one MAC-level queue. Similar arguments like already given against the first solution can be used against the second, though: providing differentiation between station priorities seems not easily possible. The advantage of VFQ over the solution 2 is the backtracking of the past, already granted WLAN bandwidth, while trying to limit the future TCP sending rates may yield unexpected results.

9.5 Discussion and Future Research

While the other ideas to solve the fairness problem of TCP over a wireless type link as described above are artificial, the VFQ service discipline provides a very natural way to control TCP flows at the base station. It is not only easily extensible to provide differentiated service (through different weights) to wireless stations, but also easily adapts to changing network conditions and different types of wireless networks.

The virtual flow approach provides also improved security compared to a two-way

[27]But a vague proposal to implement dynamic rate controller is suggested.
[28]In fact the queue size should be dynamically adopted to the number of TCP flows.

type scheduler observing the up- and the downlink direction of a wireless cell directly. Since any authorised participant of a 802.11 WLAN cell can capture[29] but also inject arbitrary traffic onto the wireless channel, he (the eavesdropper) may easily inject packets[30], which spoof the TCP communication of other wireless nodes, for example inject faked large TCP acknowledgement segments which could confuse a two-way type scheduler. Therefore a two-way scheduler provides to an eavesdropper a more effective attack vector for a `denial of service` (DoS) attack at other wireless stations. The VFQ service discipline minimises the risk of a DoS attack by observing only the downlink direction in a wireless cell, where eavesdropping is more difficult (but it is not fully impossible, since an evil user may still sniff wireless traffic and then forward some packets or just information extracted from the packets he has observed to an outside station on the Internet, which in turn may then generate spoofed TCP segments causing a similar DoS attack on the flows from other wireless stations. However, this is obviously more difficult to accomplish (keyword: IP egress filering and spoof protection on largescale Internet [118]).

Another advantage of the virtual flow scheduling approach over other solutions is its easy deployment in the situation, where the uplink and the downlink direction of a virtual link can not be observed at a single physical node. For example in Germany a service called Sky-DSL exists to provide DSL-like Internet access where copper-based DSL[31] is not available. Sky-DSL works by using a normal analogue or digital phone line (ISDN) for the uplink direction, while using a normal digital satellite card on a dedicated satellite channel for the downlink direction. Such hardware setup may impose principal problems, since the two half-duplex paths are physically separated at some point and one of them routed to a satellite station (where downlink rate and speed information may be available) and the other to the regular terrestial phone backbone.

The VFQ scheduler can be also easily extended to the case of location dependent MAC rates (e.g. to provide improved service levels to users experiencing low signal quality due to a higher distance from the base station), if such information is available at the MAC layer and can be read from the card's hardware. In this case, the VFQ cost function can be modified to incorporate a variable service rate of each participating wireless station. The HostAP Linux kernel driver for example provides a last-known MAC rate information for each destination MAC and this seems to be a common feature of actual WLAN chipsets. Of course here a few heuristic assumptions about the wireless stations must be adopted for this extension of VFQ to work properly:

- since the MAC layer can only provide historical rates based on the past transmission of packets, it must be assumed that the per-destination MAC rate does not change very rapidly - the 802.11 MAC adapts to a decreasing signal quality by executing a fall-back algorithm, which degrades the per destination send-

[29]If the cell is secured by WEP-type encryption he will be also in possession of the cell key.

[30]the AP is not able to determine the real, physical source of a packet, if the source MAC address is spoofed, for obvious reasons, in contrast to some wireline networks.

[31]The reason are missing copper lines which were replaced by optical fibers after 1990, which can in theory provide thousand times the throughput of copper, but where DSL-like access technology is not yet available at reasonable prices.

ing rate gracefully with a time resolution in the order of one second (in actual 802.11 products)

- care must be taken about the upload and download direction speeds: the AP will always know the rate it used to send a packet for a particular destination, however it will also only know the rate which a wireless node used the last time it sent a packet which successfully arrived at the AP, but in the case of a TCP packet burst, the known rate may not exactly correspond to the PHY rates of each particular packet in the burst. Therefore it would be desirable to have a history of packet rates configurable in length as seen from a particular station by the access point.

While the advantages of VFQ are numerous, there are also few drawbacks which must be mentioned too. First, the VFQ architecture is relatively complex[32] being a layer-4 scheduling solution. Tracking TCP flows has been proven to be a non-solvable problem under certain scenarios (see [107]), however in a wireless environments some of the constraints imposed on TCP flow tracking are low. In particular, the wireless network where the VFQ discipline is going to be used will have only one hop, therefore the interpretation of the IP's TTL field is straightforward (in a general flow tracking problem a tracing node may not be able to decide if a TCP packet will or will not reach its destination because of a low TTL value).

Second, since the actual wireless technology is still rather low-speed[33], the interpretation of the TCP's announced window size field in the TCP header is easily possible (this is not a strict requirement for VFQ itself, however a clean VFQ implementation must track the TCP sequence numbers and the current sender window for security reasons). Furthermore, additional research must be performed to investigate VFQ behaviour in a mix of different TCP variants. An online heuristic algorithm could be incorporated into VFQ to guess the most probable TCP version used by an observed flow. A good work in this area can be found in the interesting paper [103] investigating the problem of inferring the TCP version from packet traces.

On the other hand VFQ's algorithmic complexity is rather low (since the classical WFQ queueing can be efficiently implemented with $O(n \cdot log(n))$ complexity), but it optimally requires floating point computation, which is for example not directly available in the Linux kernel. Moreover, VFQ may fail to work properly in a situation, where TCP segments are fragmented by IP (or encapsulated in packets from a VPN tunnel, in each case the TCP header is not available to intermediary nodes), but this is not the case for normal TCP which will use PMTU discovery to prevent fragmentation[34].

More work must be done to investigate the optimal structure of the VFQ node schedulers ("black boxes"). Another open issue in VFQ is the exact formula for the cost function solving the problem of the estimation of the number of active nodes. However, as the simulation and the measurements confirmed, the error made by

[32]Not to be confused with the algorithmic complexity.

[33]Low-speed compared to gigabit-ethernet for example.

[34]the Linux TCP/IP stack is capable of performing defragmentation of all IP datagrams if configured with a special kernel option.

potentially overestimating the number of backlogged wireless stations is acceptably small.

More serious concerns could arise about the correct VFQ operation on a high-error wireless link, since the VFQ cost function considers only the TCP goodput assuming that the per node collision- and packet loss rates are equal for all stations. This does not always fit a wireless scenarios well, however, if MAC level information about packet transmission and retransmissions is available, it could be incorporated into the VFQ cost function as well, in order to provide real channel time utilisation fairness at the MAC level. Without this modification the VFQ server is likely to behave like the ELF scheduler with unlimited maximum per-station effort (but still trying to resolve the download-upload asymmetry problem while ELF doesn't). It is not easy to define, what the notion of fairness means in such scenario: should users in an airport hot-spot for example, be provided with equal application-level bandwidth, that is with equal experience of the WLAN performance independently of their physical location in the hotspot area (that is the wireless signal strength), or should they be provided with equal shares of the physical channel capacity (which is an abstract and non perceivable quantity for the majority of WLAN users) but a different end-to-end performance experience? The answer may depend on the particular application and utilisation scenario, however I believe that in practice the first idea is going to gain more acceptance from end users.

In a future continuation work a delay-fair TCP scheduler based on a different cost function and the equalisation property of the underlying WFQ discipline should be developed. An interesting approach to the TCP-delay problem has been presented in the publication [40], where the authors study a heuristic online as well as an offline algorithm for dynamically delaying TCP ACKs, in order to minimise TCP session delay.

Chapter 10

Towards Improving TCP Congestion Control

10.1 Introduction and Problem Statement

In the remaining chapter of this work I want to develop a new idea for improving TCP congestion control taking into account the previous lessons learned from the observed behaviour of TCP over wireless links. The aim however is not to provide a solution in any aspect better and superseeding all other existing TCP algorithms, but to propose another, one more solution based on a new idea. The aim is further to show that for a simple wireless network topology and similar utilisation scenarios like those used in the previous chapters, the so modified TCP algorithm is able to perform sufficiently well.

The lesson number one learned in the previous chapter states, that the state of the art Newreno TCP is not necessarily a fair sharing solution with respect to the per-node throughput, especially its short-term fairness behaviour may be rather poor. Depending on the particular network topology a TCP sender must operate, the exact link capacity, the path RTT, the permanent congestion level and the number of concurrent TCP connections (even without taking routing dynamics and similar elements into account), classical TCP[1] may exhibit a variety of operation regimes and it may be difficult to predict its behaviour in a given situation.

The second observation teaches that existing TCP algorithms exhibit a highly aggressive sender-side behaviour and can significantly contribute to network congestion, for example on a relatively slow wireless link, while in theory they should avoid to induce significant network congestion. In fact, it can even not be spoken about congestion avoidance in case of TCP Reno, because the algorithm depends on periodic link queue overflows in order to estimate the available network capacity. This implies that a single unbounded Reno connection crossing an unloaded network path, will with high probability (on average) fill the bottleneck queue to about the half. On the other side of the TCP universe there is TCP Vegas deploying a novel congestion avoidance scheme, which really avoids to induce unnecessary network congestion simultaneously being its key disadvantage in the case, that Vegas TCP

[1]**With classical I mean Reno-like TCP.**

must compete with another flows of aggressive TCP-Reno type.

To look at the congestion control and avoidance problem from a non-conventional point of view, it can be said, that classical TCP Reno operates without any notion of bandwidth during its congestion avoidance phase, that is at the 0th order of path capacity estimation and just oscillates around some constant, which is (somehow) related to the buffer space availability along its path, slowly growing but then cutting the congestion window to half, once the network path has been overloaded. On the other hand, TCP Vegas' path capacity guessing uses a target (or expected) bandwidth to estimate and classify its actual sending rate and then reacts accordingly, if the actual sending rate differs from what it expects to be, that is, Vegas TCP uses a kind of a first order estimation of the available bandwidth using the rate variables directly. Such algorithm however, is sensitive to the choice of the absolute operating point, that is to say is sensitive to the value of `BaseRTT` and as demonstrated in the previous chapter through simulation, may subsequently lead to long-term unfairness between flows sharing same bottleneck link.

Thus it would be desirable to have an algorithm, on the one side insensitive to the absolute value of the available bandwidth and on the other side, sensitive to instantaneous changes in the value of available bandwidth. This objective leads to what I call `differential congestion control` and which I want to discuss in the next section in more detail. Besides these two requirements, a TCP algorithm must inevitably address numerous issues, namely:

- a TCP sender must dispose of a cold-start[2] procedure to adapt its sending rate after either a period of inactivity or at the beginning of a new connection and must be able to find and maintain an optimal working point (size of the congestion window that is the number of packets in fly)
- it must also incorporate a capacity probing algorithm once a working point has been found but at the same time be able to avoid and properly detect any congestion build up
- it must adequately react to congestion building up in the network but also be able to detect changing network conditions like sudden capacity shortage e.g. related to connection rerouting
- and last but not least, it must be able to obtain its fair share of the network capacity in competition with other TCP flows sharing the same or parts of the network path

With the TCP-D approach I try to address the first three above points while the last being so far an open issue. The last problem is even not easy to define since in some situations it is not clear what a fair share is likely to be[3]: for example for a single bottleneck link shared by two TCP connections, one experiencing a RTT ten times bigger than the other, should "fair sharing" be defined by the fair share of available bottleneck buffer capacity or rather by equal bandwidth obtained by the

[2]I prefer this expression over "slow start" since the second is rather misleading.
[3]this is related to the notion of fairness as in ELF [41] for wireless networks.

two flows? In a more complex topology with multiple, potentially congested links and random background traffic, where TCP flows may share parts of the network path with each other[4], the definition of fairness seems to me to be a philosophical issue.

10.2 Presentation of TCP-D

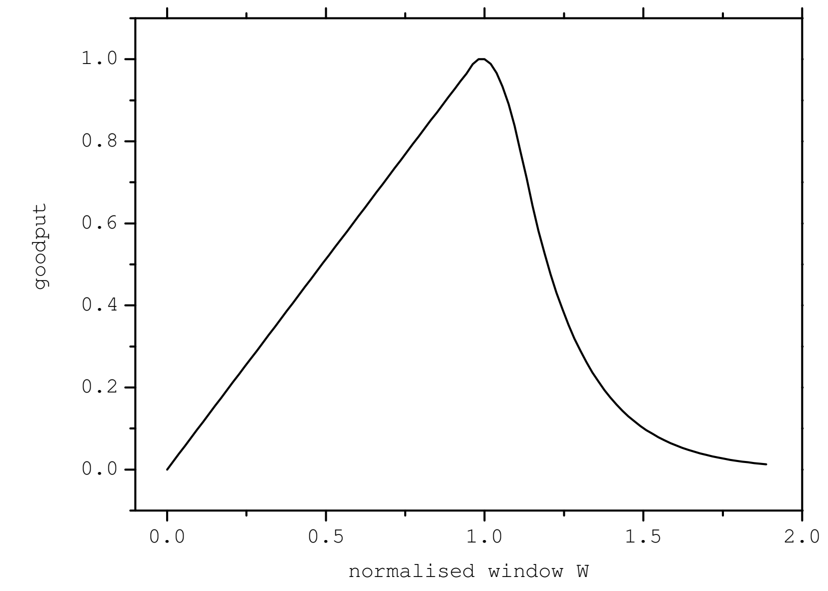

Figure 10.1: Typical TCP goodput curve over the normalised congestion window (W=1: congestion window size equal to bandwidth-delay product).

The key idea of the TCP-D approach comes from the observation of a non-congested bottleneck link once a new TCP connection appears on the non-congested network. What would a TCP sender completely unaware about the network topology and its capacity observe, once it must commence with the transmission of packets into the network and if he assumes that performing a cold-start procedure like any other TCP variant is a good practice? The sender starts with a window size of (for example) one packet and just sends its first data packet and receives the corresponding ACK[5], then he can double the congestion window and send two packets in a row, and so on and so forth as long as the corresponding ACKs are returning. At the same time he can observe a growing receiver side rate calculated by counting the number of bytes sent and acknowledged by the receiver between two reference times divided by the time difference.

However, at some point of this process the sender will observe that increasing the effort, that is the sending rate, does not yield an equivalently high increase in the

[4]**something like this is usually just called the Internet...**
[5]**Neglecting the SYN handshake.**

receiver side bandwidth as indicated by the stream of the returning ACK packets. As long as the bottleneck link in the path is not congested, increasing the sending rate will definitively increase the observable receiver side rate, however once congestion starts to build up, increasing the sending rate will rather decrease the perceived receiver-side bandwidth, since the growing queueing delay will add to the total RTT of the path and let the sender observe a decreasing effective receiver side rate. In other words, the idealised goodput curve as experienced by a TCP source has the form as presented in the figure 10.1. A similar observation and postulation for the shape of the goodput curve has been described in the paper [24]. The normalised window value of 1 on the x-achsis represents the operating point, where the TCP sender has its congestion window set to the value for which it obtains the fair-share of the bandwidth (that is of the on average available bandwidth in the network path), that is its window size is set to the effective bandwidth-delay product of the network path[6].

The new idea of my approach is its differential character. Unlike TCP Vegas which just uses the absolute difference between two quantities: the target and the actual bandwidth, that is $\Delta B = B_{target} - B_{actual}$, TCP congestion control can also exploit the bandwidth differential:

$$\xi = \frac{\partial B}{\partial CW} \tag{10.1}$$

where B stands for the receiver side bandwidth and CW for the congestion window size, therefore becoming independent of the absolute values of the expected and the actual bandwidth. In other words it is possible to exploit "infinitesimal"[7] changes in both variables and compute the goodput change, that is, to "rattle" a little bit at the sending rate and observe the response coming back from the network. The TCP sender must therefore slowly but continuously raise its sending rate and observe if the receiver side bandwidth grows or decreases. This approach leads to an algorithm which I call TCP-D.

From another point of view as shown in figure 10.2 the TCP-D idea can be interpreted as a generalisation of the rigid and rather unadaptable window probing of TCP Reno, whose well known sawtooth-like window development is nothing else than an imperfect approximation of a continuous window probing curve. In my opinion, Reno-TCP window probing during congestion avoidance is an imperfect approximation of a harmonic oscillation, which is the base principle governing a feedback-regulated system[8]. In other words I claim, that it is possible to construct a feedback based congestion control algorithm governed in its core on a sort of a wave equation:

$$\psi + \omega^2 \frac{\partial^2 \psi}{\partial t^2} = 0 \tag{10.2}$$

[6]If the bandwidth-delay product of the path is K, then the normalised window is defined by $W = CW/K$, where CW the congestion window of the TCP sender.

[7]This leads to the question what means "infinitesimal" in a packet based system...

[8]but this should not mean that all feedback regulated systems must be harmonic. For a harmonic component to appear it is required that the "reaction" of the system is proportional to its "distortion", e.g. like for a spring pendulum with force equal the displacement multiplied with a constant.

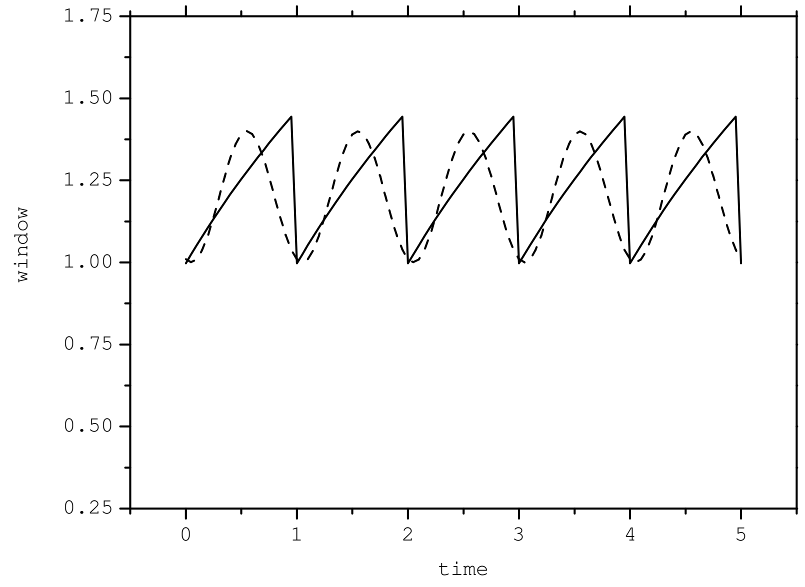

Figure 10.2: Generalised TCP window probing: Reno-like sawtooth (solid curve) and a perfect harmonic oscillation (dashed curve)

where ψ stands for the system amplitude (window size, sending rate, etc.), ω for the system frequency (regulation or oscillation frequency) and t for the system time (the physical time elapsed, the ACK or packet number, etc.). Harmonic oscillation is a process widely present in nature (simplest examples: a pendulum or a string) and occurs, if a system variable can be put in above relation to its respective second derivative (e.g. force equal acceleration). This is the intuitive motivation for introducing a congestion control algorithm based on the proportionality of the window change to the respective derivative of the measured bandwidth.

While Reno TCP lowers its sending rate if it is already too late (packet loss indicated by duplicated ACKs), a bandwidth sensitive algorithm can exploit the change in the receiver-side bandwidth response to changes in the sender-side congestion window size (that is, changes to the sending effort). Both approaches have their advantages and disadvantages, because it is not easily possible to know the exact reason why the network responds with a packet loss or a falling receiver-side rate and in some situations it would be more wise to use the first of the two possibilities but sometimes the second may yield a better result. For example, Reno TCP may cut its congestion window to half if some packets are lost (and detected by DUP-ACK) for a reason other than congestion from TCP sources (e.g. a transmission error) or if the average per-flow bottleneck buffer capacity changes suddenly (a non-responsive, high-bandwidth UDP flow arrives). Similarly using bandwidth changes as response to sending rate changes is sensitive to rapid variations in the perceived delay and probably doesn't work well for all possible network conditions. However, since the TCP congestion control problem doesn't seem to have a general, always working and good solution, any novel idea improving TCP performance and behaviour in a particular situation can be considered as a step in the good direction.

A TCP-D sender deploys a generalised cold-start procedure as follows. The initial window size is set to two segments (a value also used in Vegas TCP) and for each acknowledged segment (not for each ACK packet, this is the byte counting technique [5]) the congestion window CW is grown by the segment size multiplied by a constant factor chosen here to be 0.5. This choice resembles the behaviour of Reno during its slow-start in combination with a DEL-ACK receiver, while being less aggressive than a regular Reno-to-Reno transfer. No classical slow-start threshold is used to detect the end of the cold-start phase. Instead the cold-start phase ends, once the ξ value (as defined below) becomes negative and the congestion avoidance phase begins (which is a similar behaviour as exhibited by TCP Vegas).

In order to compute the value of ξ TCP-D maintains two counters called `target_ack` and `target_pkt`, which are both initialised at the beginning of the connection to the current ACK sequence number plus the initial CW value[9], respective to the number of data packets sent plus $CW \cdot 1.5$ [10]. Each time the target variable initialisation procedure is called, the current time is saved in the `start_t` variable, the actual CW size in `start_cwnd`, the actual ACK number in `start_ack` and the (so far) measured receiver side bandwidth in `start_bw`. On the reception of a new ACK packet the target values are checked and if either the so far received ACK number or the number of segments sent exceeds either the `target_ack` or the `target_pkt` counter respectively, ξ computation is performed according to:

$$\begin{aligned} end\_bw &= \frac{last\_ack - start\_ack + dup\_ack}{actual\_time - start\_t} \\ \xi &= \frac{end\_bw - start\_bw}{\Delta\, CW} \end{aligned} \tag{10.3}$$

where $\Delta\, CW$ stands for the growth of the congestion window between the two reference times `actual_time` and `start_t`, taking the temporary CW inflation during a potential fast retransmit into account. This is of course a rather imperfect estimation of the exact value of ξ but I will give a better idea later in this chapter. In practice TCP-D also uses an EWMA-smoothed value of the receiver side bandwidth rather than the direct sample, with configurable smoothing parameter ϵ:

$$B = B \cdot \epsilon + B_s \cdot (1 - \epsilon) \tag{10.4}$$

where B_s stands for the value of `end_bw` from the above equation and the differential is computed using the smoothed values of B at the two reference times `actual_time` and `start_t`. This makes the algorithm less sensitive to rapid but non-correlated changes in the receiver side bandwidth while being still able to detect congestion build-up before packets are lost[11]. The right choice of the value of ϵ is an open issue yet, however I have obtained good results for ϵ being between 0 (direct sample) and

[9]In practice this value is limited by another configuration variable, so that the ξ computation is performed after at least k rather than after CW packets (that is after $min(k, CW)$ ACKs), in order to avoid too rough estimation of the differential.

[10]This is so far an arbitrary choice.

[11]Of course I do not doubt that it is possible to find a scenario where the algorithm may fail in this sense.

0.95. This is a similar problem to the issue, which exists in TCP Westwood [29],[51] and whose effectiveness in the estimation of the long-living bandwidth component depends on the setting of certain filter parameter values[12].

The sender side congestion window size CW is incremented during the cold-start phase by a fixed value per acknowledged segment and the increment is reduced to $1/CW$, once the congestion avoidance phase has been entered. This choice is compatible with Reno's congestion avoidance behaviour, however I think that increasing the CW size by some (yet unknown) fraction (percentage) of the actual CW value could yield better results. For more details on this issue especially for high-speed networks see the scalable TCP paper [79].

Once the estimated value of ξ becomes negative, TCP-D reduces its sending window in proportion to the measured bandwidth change as:

$$\begin{aligned} q &= \frac{CW \cdot \xi}{B} \\ CW_{new} &= CW \cdot (1 - q) \end{aligned} \tag{10.5}$$

where q is a dimensionless quantity[13] and can be therefore used for an AIMD-like window control. However, the TCP-D algorithm restricts the value of q to a maximum value named Q[14] to prevent too radical [15] CW reductions. The rationale for limiting the q value in TCP-D to less than the classical Reno cutoff value of 0.5 is as follows: while Reno TCP detect congestion by packet losses very late and then it must definitively cut its congestion window size rather radically to half, TCP-D claims to detect congestion before the queues in the network actually overflow, so that such a strong window size reduction is not really justified. Once the congestion window has been reduced, the new additive window increment is calculated as $1/CW_{new}$.

Further, the TCP-D sender performs similar fast-retransmit and recovery procedures as classical Reno TCP and also reduces its congestion window size, once three (or a configurable amount) duplicate ACKs have been received, however the reduction is also limited by the same Q factor introduced above. The argument is similar to the case of a negative estimate of the ξ value: since the primary congestion detection mechanism exploits the sign of the expression for ξ, packet losses due to self-induced congestion are supposed to be rare events, therefore being rather the result of congestion caused by other parallel TCP flows at the bottleneck queue. Thus reducing the congestion window to half seems too radical. An additional modification with respect to the window control behaviour of Reno TCP is introduced into TCP-D in order to improve upon multiple packet losses from one window of data. Once three duplicate ACKs have been received, the window size is lowered to $(CW) \cdot (1 - Q)$ and a target acknowledgement number is stored in `target_reduce` variable as the actual highest acknowledged segment number plus the reduced congestion window size and further window reductions are suppressed, until the `target_reduce` ACK

[12]**See for example the NS2 Westwood modules WWW page for explanation and meaning of the filters.**

[13]**As can be seen by evaluating the units of the respective expressions in the fraction for q.**

[14]**Which is set currently to 0.25 in my NS2 module.**

[15]**which may also result from the simplistic computation of the derivative.**

is received[16].

The timeout behaviour of TCP-D resembles Reno TCP and implies the cold-start procedure once the retransmission timer expires. Under normal circumstances, that is with some buffer space available at bottlenecks and stable RTT in the network, timeout events are rather rare in TCP-D since the algorithm is trying to keep the sender-side congestion window size close to a suitable, non-overshot value.

10.3 Implementation and Validation in NS2

This remaining section provides initial simulation results for TCP-D in the simple wireless network topology 7.1 already used in previous chapters. While it would be also interesting to start with a wired scenario and study the behaviour of TCP-D in a more general LAN/WAN topology, the wireless simulation has been chosen for convenience and comparability with previous simulation results. The TCP-D sender modification has been implemented as an additional module in NS2 and based on the code of the NS2 Reno module[17]. Because of the complexity of NS and especially its TCP modules[18], I do not claim that the implementation is complete, nor that it is mathematically a very accurate implementation of the initial idea of differential congestion control. However, my logical argument for the convenience of the differential control idea bases on the simulation results presented below, since I do not believe, that a rather approximate implementation of a bad idea can yield any good result. During the practical work on NS' TCP modules I experienced rather the opposite thing while initially trying to change the existing code of Reno. Even small but "stupid" changes to the code virtually always yielded an algorithm which did not work at all, that is for example suffered from huge amount of retransmissions or not able to keep enough packets in the network pipe. Therefore I strongly believe that the differential control idea has the potential to provide a novel direction in the area of TCP congestion control research.

The first simulation run was intended to observe the behaviour of TCP-D only connections in direct competition for the wireless bottleneck link capacity. The figure 10.3 shows the transferred volume over time and the corresponding Jain's fairness index (with the base interval used to compute the index value set to one second) for 6 parallel TCP-D flows (the 6 curves perfectly overlap in the figure!). The difference in the observed short-term fairness with respect to the fairness of 6 Reno-TCP connections (figure 7.2) is enormous and can be attributed to the finer-grained[19] window control of TCP-D, which has been also depicted in the figure 10.4. The evolution of the sender-side congestion window size has been plotted over the simulation time (from 0 to 65 seconds) and the zoomed initial part of the curve illustrating the cold-start phase shown in the upper part of the figure 10.4.

Unlike in the Reno TCP simulation, window synchronisation between the flows does

[16] It may require a modification in the case of highly and frequently changing network bandwidth.

[17] This is due to the low code complexity of the Reno module.

[18] NS TCP is based on a class hierarchy with a complex base class and simpler derived "congestion control" classes like Reno or Vegas.

[19] On a finer time scale.

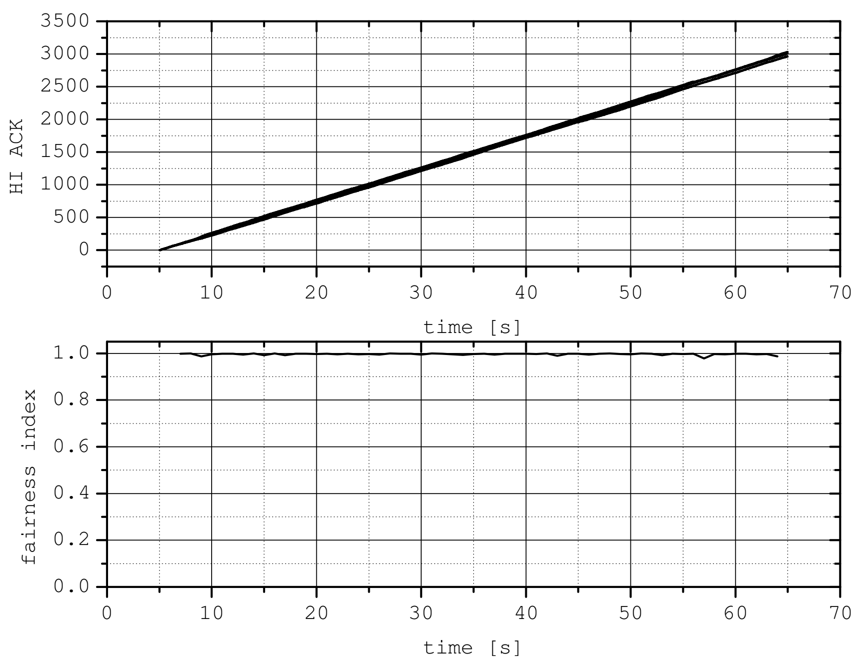

Figure 10.3: Scenario 1: six TCP-D downloads, transferred volume (upper plot) and the Jain's fairness index (lower plot) shown.

not take place for the same per-station and AP queue lengths and same network topology. The observed window oscillations of TCP-D are just of "infinitesimal" order compared to the available buffer space at the bottleneck, while in contrast Reno TCP oscillates in the whole queue space[20]. During the complete simulation time there are just two timeout events (at about 15s and 56s) where one of the sessions must recover from the loss by another dynamic cold start. The two timeouts could be even avoided, if the the base code of the NS2 TCP class had been changed, because it uses the estimated RTT value but also the variance of the RTT[21] to program the retransmission timer. Since in TCP-D the congestion window size remains much more stable than for Reno, the long-living estimate of the RTT variance is rather small and once an unfavourable constellation of queue length, WLAN backoffs, etc., delays some of session's packets, a timeout may occur, which would not happen for the same delay variation in case of classical Reno-TCP with its huge RTT variance.

A look at the queue occupancy plot in figure 10.5 confirms the above stability result and clearly shows, that the bottleneck queue size remains very stable and exhibits just small but rather random oscillations, therefore providing much more stable delay to the end hosts. This is an important issue for multimedia applications, which may even adapt to changes in the available bandwidth but can be very sensitive and unhappy about variations of bandwidth and delay, once a media transfer has been started at a certain speed and an estimated QoS level. This sensitivity experience is

[20]**As long as no flow control from the receive side limits the maximum reachable *CW* value.**
[21]**I must admit that at this point the NS code is not easy to understand.**

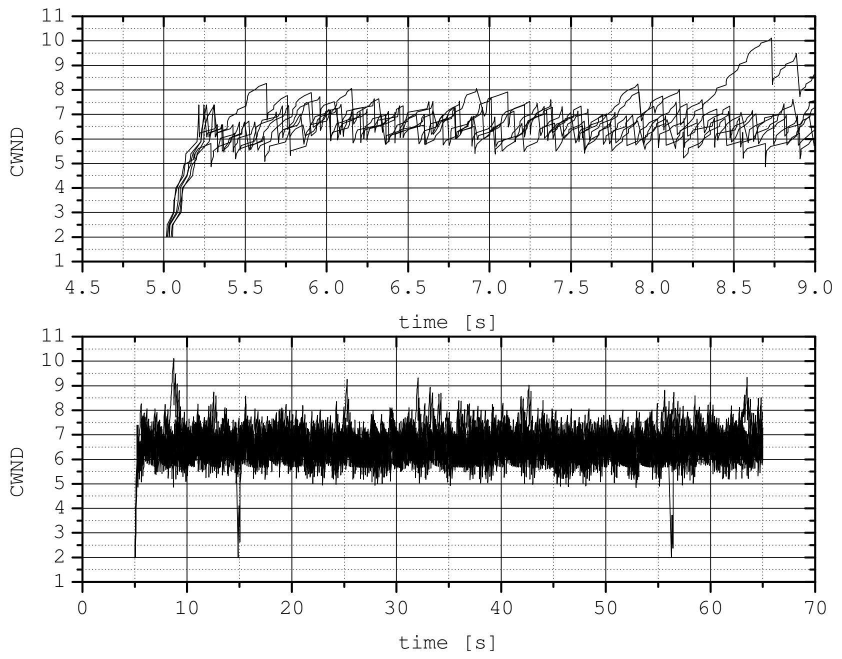

Figure 10.4: Sender-side congestion window size development for scenario 1, upper graph shows a zoom at the cold-start phase.

known to for example users of MP3 streaming over TCP[22], who frequently experience interrupted music "pleasure", once the TCP connection used for media transfer must recover from packet losses by a timeout.

To further study TCP-D behaviour and provide an experimental argument, that it is not harmful (or at most as harmful as the other existing TCP sender side congestion control algorithms) to the network, a comparison with other TCP versions must be made. This is rather a huge task and would be out of the scope of this chapter, if a complete comparison of "everything with everything" should be done (for example SACK on the receiver side, TCP-D on sender side, in competition with Reno, Newreno, Westwood, Vegas, then DEL-ACK on the receiver side with different settings for delay, then standard ACK receiver, etc. etc..., changing network delay, queue size...). In other words I just want to provide some basic and simplified scenarios to study the cooperation of TCP-D with other TCP "colleagues".

For this purpose two more simulations to analyse a direct competition situation of TCP-D with NS2 Reno TCP have been ran. In the first of the two simulations there are five bulk TCP-D sessions starting their data transmission at the time $t = 5$ and an additional Reno connection arrives 30 seconds later. In the second simulation scenario, the roles of Reno and TCP-D have been exchanged. Remember that a similar simulation (figure 7.6) has been presented for Vegas and Reno in the previous part of this work and Reno TCP was found to aggressively outperform TCP Vegas(2,4).

[22]See for example the shotcast.org Internet radio homepage.

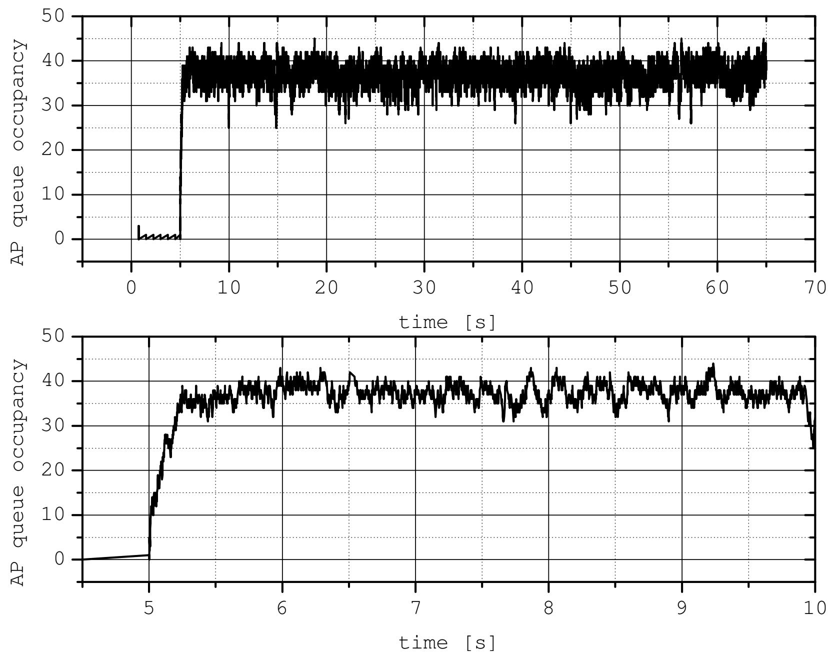

Figure 10.5: Queue length development in scenario 1 at the access point.

The result for TCP-D is presented in the figure 10.6 and there is a clear improvement in the resulting bandwidth shares compared to the Vegas-Reno simulation, however, TCP Reno still outperforms the other TCP-D sessions and this is not so surprising, since on the average, the congestion window of Reno will periodically grow until the bottleneck queue is full, while TCP-D tries to avoid packet losses at the bottleneck queue. Thus TCP Reno (including other Reno-like TCP algorithms e.g. TCP Westwood) will aggressively acquire more bottleneck buffer space than any, queue overflow avoiding scheme like TCP-D, resulting in unfair bandwidth sharing[23] at short as well as at long time scales. While Vegas miserably fails in such competition situation, TCP-D is more aggressive (since it does not rely on a "belief" about the path parameters like Vegas but tries to test the network response to an increased sending effort) and therefore more successful in fighting against concurrent Reno flows. Also the fairness index curve shown in the upper graph of figure 10.6 demonstrates the aggressiveness of Reno, once it starts to fill the bottleneck queue. Periodic queue oscillations induced by Reno appear synchronously in the fairness index plot as periodic variations of the Jain's fairness index value.

The opposite situation with one TCP-D connection opened in the middle of the simulation has been shown in the figure 10.7 and it is important to mention, that this is a completely different situation compared to the first scenario, since the permanent congestion level in the network induced by the five Reno flows is rather severe, leading to window synchronisation and periodic bottleneck queue oscillations as explained in chapter 7. The volume-time curve in the figure 10.7 shows that the TCP-D

[23]since for a congested link fair bandwidth sharing depends on fair sharing of the available buffer space

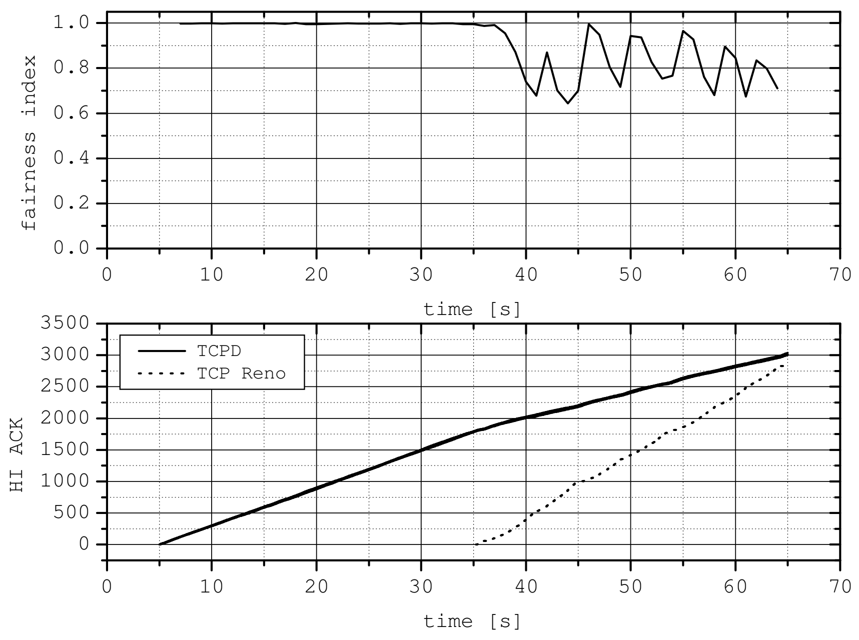

Figure 10.6: Five TCPD flows in direct competition with one Reno flow

flow is sufficiently successful in acquiring reasonable bandwidth share of a periodically congested link and receives about 50% of the Reno per-flow bandwidth. This is slightly better than a Vegas(2,4) flow in the same simulation scenario. Obviously, the fast-retransmit mechanism incorporated in TCP-D together with its differential congestion avoidance scheme is sufficiently successful in avoiding frequent timeouts even in a severe congestion situation.

The last simulation presented in this chapter has been devoted to the study of a TCP-D against Vegas competition scenario, where one additional TCP-D session arrives in the middle after 30 seconds of ongoing Vegas transfer. Looking at the result shown in figure 10.8 a relatively good compatibility with Vegas-TCP[24] can be attested to TCP-D. While TCP-D still outperforms Vegas, the resulting bandwidth distribution of roughly 33% more for the additional TCP-D session seems reasonably fair.

Finally, a conclusion about the operating point of TCP-D can be made and based on the bandwidth shares obtained in direct competition with the other two TCP algorithms. Since the average throughput of a bulk (that is always backlogged) TCP connection is proportional to the average congestion window size of the TCP sender (for same RTT, packet loss, etc.) that is to the average number of packets held "in fly" and the respective bandwidth shares B_D of TCP-D, B_V of TCP Vegas and B_R of TCP Reno obtained from above simple wireless competition scenarios can be put into the following relation to each other:

$$B_V \leq B_D \leq B_R \tag{10.6}$$

[24]**For classical Vegas(2,4) TCP long-term unfairness may result between Vegas flows.**

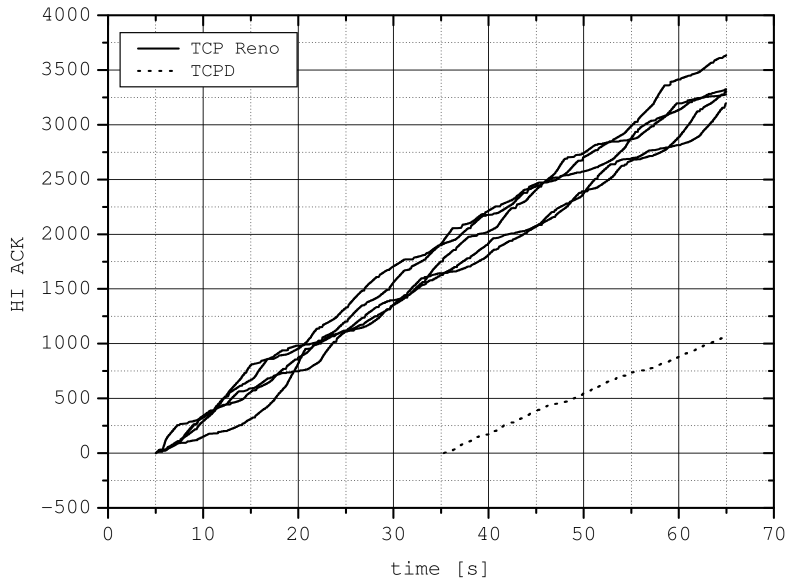

Figure 10.7: Five Reno flows in direct competition with one TCPD flow

it can be concluded, that the average operating point of TCP-D with respect to its congestion window size lies between the operating points of TCP Vegas and TCP Reno. Looking again at the goodput curve 10.1 TCP Reno's congestion control from time to time touches the congestive falling edge by overloading the network and inducing packet losses, while Vegas congestion control may stay in the linear part because it has got a wrong estimation of BaseRTT, therefore under-utilising the available network capacity. Somewhere in the middle between the two places the operating point of TCP-D can be presumed.

10.4 Conclusion and Future Work

A final conclusion about the potential of the differential approach for congestion control is not possible yet, but the results obtained through simulation suggest, that the idea investigated in TCP-D is valuable and can further improve TCP performance with respect to throughput fairness and alleviation of permanent network congestion levels. In future research a strong mathematical analysis of the approach presented here must be given, however analogically as for the mathematical models of Reno or Vegas, only a selected and well defined scenario can be taken into consideration.

An improved prototype of TCP-D must utilise higher accuracy for the computation of the bandwidth derivative and could for example use an extrapolation technique to obtain a continuous estimation of the ξ value and thus become independent of the size of the sampling interval k. Unfortunately this is a requirement, which may encounter practical limits in a real-world implementation where computing power is usually a limited ressource - unlike a NS2 simulation, where the real computation time necessary for processing of each packet is not significant. The prevalent

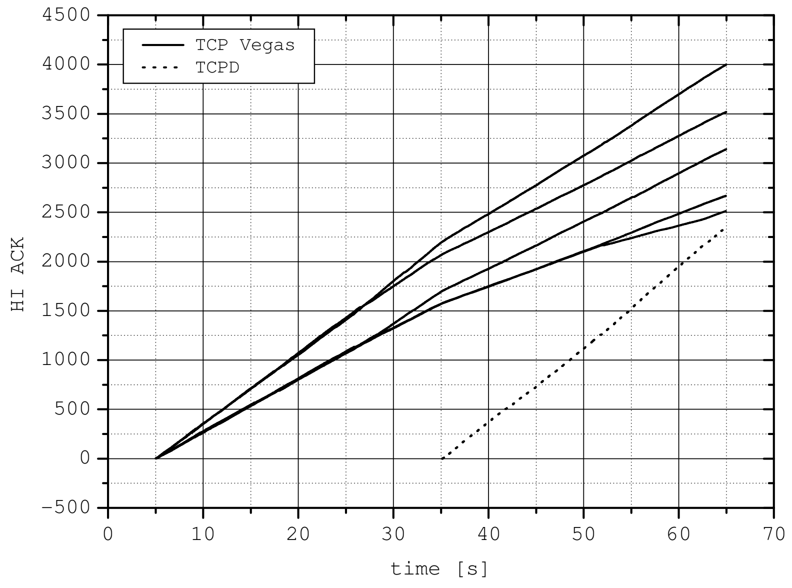

Figure 10.8: Five TCP Vegas flows in direct competition with one TCPD flow

domination of Reno-type TCP in today Internet can be contributed to its really low algorithmic complexity, its simple structure and the relative ease of implementation[25]. Improving the TCP congestion avoidance and the cold-start behaviour inevitably leads to increased algorithmic complexity and requires higher computing power in the end hosts. This observation can impose a limit on the further development in this area[26] and the acceptance of improved TCP algorithms, since the average utilisation scenario of the Internet for the majority of end-users means "download". Therefore the effort needed to perform improved and more complex congestion control is going to concentrate at those server machines providing the consumed content. One possible way out of this dilemma would be the investigation of a receiver-side congestion control, in order to distribute the computation time onto the client machines.

Second open TCP-D issue is the fair sharing of bandwidth with other TCP algorithms, however as explained in the analysis of TCP-Reno, the behaviour of Reno should not be taken as reference system. The investigation of simple scenarios performed in this work, leads to the conclusion, that classical TCP is a deficitary algorithm with respect to the short-time fairness, but as well with respect to real congestion avoidance in a few-users scenario like for example imposed by a wireless-wired integration situation. The core of the fairness problems is the aggressive window probing of Reno-like TCP, which must overflow the bottleneck link queue

[25]**However it should be mentioned, that despite this the TCP/IP stack is one of the most complex subsystems of an operating system e.g. like the Linux kernel.**

[26]**It is worth to note that for example with techniques like Gigabit or even 10 Gigabit Ethernet classical Reno TCP requires a substantial amount of computing power to be spent on checksumming and memory operation and imposes a natural limit for deployment of those technologies in consumer PC hardware. This may change with advances in hardware architectures (e.g. PCIe).**

first, in order to estimate the available capacity. Here a fine-grained congestion window control algorithm based on a harmonic oscillator model can substantially improve the obtained fairness level. This subject requires further investigation.

Chapter 11

Conclusion and Perspectives

The thesis provides an experimental analysis of various performance problems in wireless networks such as IEEE 802.11 with emphasising the end-to-end aspect. The central role in the analysis plays the concept of fairness. Because the whole issue of wireless network performance is very complex, it is reduced to three main layers (MAC, IP and TCP), which are then studied in more detail in sub-contributions and each time a generalisation of the identified core problem is presented there.

First, the distributed access method of 802.11 is studied from the viewpoint of fairness but also from the viewpoint of general structure of collision avoidance schemes, which are identified in this thesis as just a special application of (distributed) hashing and sorting.

Second, it is shown that a single-layer solution like IP-level WFQ packet scheduling in the access point for the downstream (that is the wireless side of the AP) direction is not able to sufficiently solve the fairness as well as the bandwidth provisioning problem in an dynamic WLAN environment. A generalisation of fair queueing algorithms using a cost function is presented together with a specialised application to the 802.11 WLAN case.

Third, the concept of fairness is investigated for state-of-the-art TCP, where fairness is found to be insufficient especially for networks incorporating WLAN links and user behaviour resembling average WLAN utilisation scenarios (few connections, scare bandwidth). Thus a generalisation of the well-known congestion avoidance algorithm of TCP with the aim of improving its fairness is presented. The three contributions are summarised below.

11.1 Wireless Channel Access Methods

11.1.1 Achievements

The contribution presented in this manuscript is to answer the question about the structure of an optimal collision avoidance algorithm for wireless networks. This question has been investigated in the context of two recent and interesting proposals improving the basic 802.11 access scheme, namely Idle Sense [60] and TCF [87]. My work provides an alternative view at the CSMA/CA type access schemes and relates these group of algorithms to distributed hashing and sorting. It is known that the DCF method in IEEE 802.11 is very difficult to analyse by mathematical means as shown for example in the articles [22] and [139]. The simplified model for CA presented here shows that it is possible to construct far less complex collision avoidance algorithms without any significant sacrifice of short-term fairness.

Thus a reference system called $\mathcal{H}$ suitable for performing comparisons of different collision avoidance schemes has been constructed. It uses an uniform random distribution of backoff slots over a fixed size hash table of C places. For each reasonable (that is relevant in practice) table size C and increasing number N of contending stations the $\mathcal{H}$-scheme has been simulated and the average collision probability $p_c(C,N)$ as well as the average number of empty slots (wait time) before a successful transmission $W_T(C,N)$ and before a (visible) collision $W_C(C,N)$ has been obtained.

If now another collision avoidance algorithm (like standard IEEE 802.11) is given, its statistical properties can be quantified by comparing its average $\bar{W_T}(N)$, $\bar{W_C}(N)$ and $\bar{p_c}(N)$ values (obtained in a respective simulation of that algorithm) with values from the simple hash table scheme at the point, where $\bar{p_c}(N) = p_c(C,N)$ for a fixed number of contending stations N. Simultaneously an effective statistical window size $\bar{C}$ of the algorithm in question can be defined and obtained by taking the respective C value of the $\mathcal{H}$-scheme for equal collision rate.

While the simple[1] $\mathcal{H}$ access scheme is just a construction well suitable for building a reference system, a practical channel access algorithm must further solve a trade-off problem between on the one hand as low as possible visible (that is appearing on the channel) collision rate and on the other hand low packet wait before each transmission respective collision event. This requirement means that in a probabilistic system[2] the tradeoff to be solved is the time wasted in channel collisions plus the time wasted in waiting due to idle slots before each transmission attempt. This tradeoff problem can be quantified for any arbitrary probability distribution $p(t)$ by introducing a cost functional of $p(t)$ and its respective CDF[3].

To solve the tradeoff problem the Euler-Lagrange differential equation is applied to a simplified version of the optimisation task in order to find an improved probability distribution of the $\mathcal{H}$-system. The solution delivers insights about the structure of the optimal probability distribution for $\mathcal{H}$, however the exact solution of the problem could not be found due to its mathematical complexity. Despite this difficulty the insight found in this part of my work is that in order to only avoid channel collisions it is optimal to spread the stations over all of the existing C table slots with equal probability $p = \frac{1}{C}$. On the other hand in order to optimise the total cost with respect to the utilisation of $\mathcal{H}$ as a wireless access algorithm, it is more advantageous to use slots at the beginning of the table to decrease the average packet wait delay. The trade-off between the two contrary goals yields a shape of the probability distribution close to a straight line with a negative slope.

Further result presented in this work are analytics formulas describing statistical

[1] **constant** $p(t)$.

[2] **no "hidden variables" in the stations,** $p(t)$ **not time-dependent.**

[3] **cumulative probability distribution.**

properties of $\mathcal{H}$ for an arbitrary probability distribution $p(t)$. In the generalised N station case and a given $p(t)$ the expression for the integrand of the cost functional can be written as a function of $p(t)$ and its respective CDF $c(t)$ to be:

$$\begin{aligned} F =&(A+t)\cdot\left(\binom{N}{0}p(t)^N + \binom{N}{1}p(t)^{N-1}(1-c(t)) + \ldots + \right.\\ &\left.\binom{N}{N-2}p(t)^2(1-c(t))^{N-2}\right) + t\cdot\binom{N}{N-1}p(t)(1-c(t))^{N-1} \\ =&(A+t)\cdot\sum_{i=0}^{N-2}\binom{N}{i+1}p(t)^{N-i}(1-c(t))^i + tNp(t)(1-c(t))^{N-1} \end{aligned} \tag{11.1}$$

where the consecutive terms can be interpreted as contributions (probabilities) of the different collision orders, that is to say, collisions of two, three, etc., nodes at the minimal (that is winning the contention) table slot to the overall cost, and the last term contributing for the packet transmission cost. The above probability terms can be further used to calculate the averages of packet wait times for transmissions as well as for collisions, if the corresponding transmission or collision term is weighted with the slot number and integrated over the definition domain.

Another contribution of this part of the work is to provide another, complementary statistical quantification for short-term fairness. The short-term fairness defined by any $\mathcal{H}$ (that is for an arbitrary $p(t)$) system has been found to provide a reference point for defining short-term fairness. By intuition another statistical metrics different from the Jain's fairness index has been introduced. It quantifies the deviation in the short-time fairness of an arbitrary access algorithm $\mathcal{A}$ from an equal cycle-by-cycle probability scheme like $\mathcal{H}$ and is called the self-correlation probability p_s.

The self-correlation probability index $p_s(t)$ is defined in the context of the wireless channel access problem as the probability that a station, which has sent a packet at the time T will have sent another packet at the time $T+t$, averaged over a sequence of channel access events. In practice it makes sense to define $p_s(t)$ not over time but over a sequence of wireless transmission cycles marking each of the N nodes with an unique ID $n_i, i = 1\ldots N$.

The analysis of an ad-hoc modifications to the original 802.11 DCF performed using the p_s method shows clearly, that the short-term fairness issue in wireless networks is not directly related to the structure of the backoff algorithm executed in case of collisions, neither to the modifications to the absolute contention window size performed after each (re)transmission. The relation between fairness and the backoff procedure is only indirect: by the correlation between the transmission probabilities for a given node at a reference time T and a later time $T+t$. Therefore the real source of short-term unfairness has been identified as the positive time-correlation between consecutive packet transmissions of a given station and a quantification metrics ("control knob") measured by the autocorrelation coefficient $p_s(t)$ of the CA scheme in question has been presented.

Further achievement of this work is to formalise the problem of collision avoidance and relate this group of algorithms to distributed hashing and sorting. While a sorting algorithm requires that a (partial) ordering operator "<=" is defined, collision avoidance schemes may be defined as a group of algorithms using a distributed version of the sorting operator called $\mathcal{M}$. It can be defined as the mapping $\mathcal{M} : \mathcal{S} \rightarrow C$ from the set $\mathcal{S}$ of all $S_j \in \mathcal{S}$ of the backoff timer values of each backlogged station onto the image $C = \{0, 1\}$ and defined for non-empty S_j as:

$$m_j = \begin{cases} 0 & \text{if } b_j^{i_k} = min(S_j) \wedge \forall l \neq k \;\; b_j^{i_k} \neq b_j^{i_l} \\ 1 & \text{otherwise} \end{cases} \tag{11.2}$$

where m_j stands for the value of $\mathcal{M}$ in the contention cycle j and means $m = 0$ for successful transmission event[4] and $m = 1$ for channel collision. Therefore $\mathcal{M}$ can be understood as an elementary coupling (information exchange) between contending stations and their backoff timers. Its value can be computed locally (locality) by any participating station for each contention cycle.

This formal definition of $\mathcal{M}$ allows to design a broad spectrum of distributed and cooperative channel algorithms based solely on the locally computable value of m_j, for example such minimising the expectation value $< m_j >$[5] for $j \rightarrow \infty$. For a noisy channel the expression for m_j must by augmented by a discrete random variate[6] $\sigma_j : j \rightarrow \{0, 1\}$ with an expectation value between 0 (no wireless errors) and 1 (100% loss).

11.1.2 Perspectives

While the above achievements are encouraging, there is still a number of open issues awaiting concrete solutions in the area of collision avoidance. Two most important ideas for a follow up research are in my opinion:

- derive an optimal residual backoff procedure using a time-dependent distribution $p(t)$
- derive a practical collision resolution scheme based solely on $\mathcal{M}$

The approach using probability distributions as studied in this work can be presumed to be complete only if the fully generalised case using a time dependent distribution function $p(t, t')$[7] is analysed. This idea can be viewed as an extension of the concept of the residual backoff as used in 802.11 permitting to define different notions of the word "optimal", e.g. by imposing the requirement that the per-slot and per-contention-cycle transmission probability in the virtual hash table does not change after a transmission attempt of a station (which is not the case for the original IEEE 802.11, nor for Idle Sense). Alternatively the designer of a modified CA algorithm may desire to change the short-term fairness behaviour of his CA scheme and therefore look for another (that is different from the classical residual backoff) transformation of the per-station probability distributions satisfying his needs.

[4]note that $m_j = 0$ means also for the contention winner numbered with l (that is the station which computed $m_j = 0$) that $b_j^{i_l} < b_j^{i_k} \; \forall k$ and this is the rationale for naming $\mathcal{M}$ distributed ordering operator.

[5]that is minimising the average collision rates in equilibrium.

[6]or a variate resembling the error behaviour of the wireless channel (e.g. bursts).

[7]where t stands for the definition domain variable and t' for the system time.

The second key idea to be implemented in practice is a working algorithm based on local computation of $\mathcal{M}$. I strongly believe that there is a complete class of CA-algorithms in analogy to well-known sorting and hashing. The statistical properties of each of these CA schemes (like average packet wait, collision rates and short-term fairness) will be different and may be even made tuneable if required by the application scenario. Therefore the future vision is to design a meta-CA scheme, where each of these parameters can be controlled by its respective control knob. While the existence of such meta-algorithm is not yet proven, there are "points" in the space of possibilities, namely IEEE 802.11, Idle Sense and TCF spanning a range of statistical properties. I did not find any counter-evidence to this assumption. A follow up research work on this area is actually in full progress.

To finalise the list of open issues in the research area devoted to channel access and collision avoidance algorithms I want to mention the necessity for generalising the quantification of short-term fairness as so far being usually a synonym for the Jain's fairness index J to the case, where the absolute sending probability of each contending station is not equal. This is a case relevant for example for the upcoming 802.11e with multimedia extensions, where the traffic differentiation between classes is based exactly on different probabilities for sending a packet in the respective class. Here the classical approach using J is going to fail, since it does not account for the intended "unfairness" of the system. So for example how should the situation of two 802.11e stations one sending in a high-priority class and second in a low priority class (and gaining only half of the throughput of the first on long-term) be classified? Note that the approach using p_s as presented here provides a direct and natural solution out of this dilemma and may be used as a hint for how to augment the Jain's fairness index to be a good metrics in the above situation too. This is subject of another work in progress.

Another contribution of this thesis is to open a door for a new research direction in the area of QoS in wireless multiaccess protocols based solely on probability distributions over a window of backoff slots. The classical IEEE 802.11 DCF and its successors like EDCF are just special cases of a broader class of access algorithms namely $\mathcal{H}$ and time dependent $\mathcal{H}$. Despite the fact that I did not provide a systematic study of the dependence of the resulting QoS levels (packet wait, collision rates, etc.) from the probability distribution $p(t)$ in $\mathcal{H}$, they can be derived analytically with easyness[8] using the result for the cost functional derived in the analysis of the optimal distribution function.

Future research must explore this idea in more detail. One possible direction would be to compare the achievable QoS levels for a generalised H access scheme with multiple distributions for each class to the traffic differentiation obtained in the special case of the upcoming 802.11e standard. Another goal of such research would be to analyse the scalability of resulting QoS levels with an increasing contention window size. Note that while for probabilistic[9] $\mathcal{H}$ short-time fairness is guaranteed, the opposite is most likely going to be the case for 802.11e with its current architecture.

[8] **for given** $p(t)$**.**

[9] **not time-dependent.**

11.2 End-to-End Performance in WLANs

11.2.1 Achievements

The contribution of this part of my work is to shows that just an one-layer solution like packet level scheduling at the access point in the downstream direction is not able to sufficiently solve the fairness as well as the bandwidth provisioning problem in an dynamic WLAN environment. This issue is related to the characteristics of the physical packet transmission based on the 802.11 DCF as well as to the expected average utilisation scenario of the network resources in places, where WLANs are used to provide seamless Internet connectivity.

The experimentation as well as simulation results presented in my work reveal potential fairness problems (upload-download asymmetry), if the WLAN traffic is dominated by classical TCP sources. The core of the problem is identified as different sensitivity of TCP sessions to packet drops at the access point and is prevalent if the access point has no priority access to the wireless channel while servicing TCP traffic for multiple wireless end-stations.

Second achievement on this area is the extension of the WFQ (weighted fair queueing) family of algorithms to imply a broader notion of network ressource utilisation by relating the virtual service times of QoS classes to physical observables through the introduction of a suitable cost mapping function. This modification to WFQ permits to implement a variety of different queueing disciplines providing fairness not at the observed throughput level but at the level of a different physical quantity (observable).

The extension of WFQ is then applied to the wireless case and the virtual flow queueing (VFQ) scheduler for TCP is presented in this work. VFQ is based on the wireless transmission time cost function and regulates TCP flows either by accounting for the observed data packets (download) or for the ACK stream in the opposite direction (upload). It provides a very natural way of scheduling TCP flows in those dynamic WLAN environments with the additional advantage of improving TCP security compared to a two-way type scheduler observing the up- and the downlink direction of a wireless cell directly. While the original goal of VFQ was to restore fairness in obtained capacity shares between different WLAN stations, it is also capable of providing weighted fairness if desired. Another advantage of the virtual flow scheduling approach over other solutions is its easy deployment in the situation, where the uplink and the downlink direction of a virtual link can not be observed at a single physical node.

11.2.2 Perspectives

Despite above achievements the open issues which could not be sufficiently solved in VFQ are:

- the structure of the station schedulers ("black boxes") providing improved QoS provisioning for flows from a single station
- integration with the 802.11g standard but also with other extensions of the orig-

inal 802.11b standard e.g. like the upcoming multimedia extensions of 802.11e

- adaptation to location dependent transmission speeds and dynamically changing per-station packet error rates
- study of the behaviour of the VFQ solution for algorithms other than classical Reno-TCP

While the goal of providing per-station channel fairness could be achieved in VFQ, the choice of round-robin schedulers for servicing flows from a single destination does not permit real QoS provisioning on a per-flow basis. The VFQ discipline presented in this work only guarantees that each wireless station can on average benefit from equal channel transmission time, however this may not be the right policy in presence of real-time constraints like required for VoIP[10]. Note that the goal of VFQ was even not to solve a global real-time provisioning problem, but only the fairness problem as observed for TCP (layer-4) and cannot be therefore considered as principal limitation. VFQ is just one solution applicable in those ad-hoc and hot-spot scenarios. Nevertheless it may be still desirable to have some amount of control over the traffic differentiation between flows for a particular station and I believe that replacing the simple round-robin server by a weighted round robin discipline can provide a satisfactory solution, but certainly more research must be performed here.

The integration and adaptation issue with the successors of the 802.11b is twofold. On the one hand the increased PHY rates provide faster payload transfer rates, but on the other hand the original DCF access method is adopted without modifications. It is out of the scope of this work to deliver a comprehensive comparison between 802.11b and 802.11g and evaluate the second standard, however two things should not go unmentioned: up-to-date 802.11g products usually provide a compatibility mode (802.11b/g mixed), where only the PHY transmission rate and the modulation type is changed (packet bursting may or may not be available here), but also a 802.11g-only mode, where also the timing[11] during the backoff procedure as well as the PLCAP header parameters are different from 802.11b.

The compatibility mode is obviously a case where VFQ queueing may improve the performance of a 802.11g network due to its even more increased single transmission overhead. The second genuine 802.11g operation mode can be assumed to suffer from similar unfairness problem like for the older 802.11b, since the procentual overhead has not been significantly lowered with respect to the increase in the PHY sending rate. On the other hand the packet bursting mode included in 802.11g must be considered in the VFQ cost function and may be used for a seamlesser integration with TCP, if packet trains (bursts) for a single stations are encountered.

The decision for breaking down TCP packet bursts in VFQ was the inability of the original 802.11b MAC to map an upper-layer burst onto a single MAC level burst decreasing the overhead imposed by DCF, so that there was no real advantage by permitting one station to send or receive a TCP burst. However if the VFQ cost

[10]and VoIP seems to be the driving force for actual developments in the 802.11 area (like multimedia extensions etc.).

[11]short slots.

function is augmented to correctly account for the cost of a packet burst, the 802.11g bursting mode may be easily integrated into VFQ. At this point a tradeoff problem must be solved again, because sending more than one packet to a destination may increase the visible cost e.g. in case of sending multiple TCP ACKs in a row (therefore the end host may flood the wireless network with new packets from its actual TCP sending window)[12], but on the other hand it may decrease the total cost, since multiple DCF contention cycles can be avoided.

An intelligent practical solution for the idea of integrating the bursting mode into VFQ must be further investigated and will require a look-forward cost function taking into account not only the "exposed" packets (that is the next packet to send from a node "black box"), but also some amount of other (consecutive) waiting packets. An integration with 802.11g hardware will also require higher amount of cooperation than for 802.11b e.g. to determine whether packet bursting (e.g. for a given destination) is available at all and whether the burst from the upper layer has been sent as intended[13]. It is worth to note that an suitable application[14] of the generalised WFQ scheme is capable of providing a minimisation of the total channel overhead e.g. by intelligently distributing packet bursts in 802.11g. This should not be confused with the principal equalisation property of WFQ with respect to per-class virtual service times.

The integration with MAC-level QoS differentiation may be worth some investigation too, however the properties and the behaviour of those extended 802.11e products are still unexplored yet. There are just few product lines providing those extensions at the moment of this writing (e.g. the Linksys WRT WLAN router line based on a Broadcom chipset) and I think that those devices should be evaluated first before any other thoughts about a potential integration with VFQ can be made.

The VFQ scheduler can be also easily extended to the case of location dependent MAC rates (e.g. to provide improved service levels to users experiencing low signal quality due to a higher distance from the base station), if such information is available at the MAC layer and can be read from the card's hardware. In this case, the VFQ cost function must be modified to incorporate a variable service rate of each participating wireless station. The problem is not whether the accounting can be done or not, the problem is just the availability of suitable rate (and error) information from the MAC layer.

In a future continuation work the idea of generalising the WFQ-family of fair queueing algorithms through the mapping of class virtual times onto a physical observable by using an appriopriate cost function must be studied in more detail. For example a delay-fair TCP scheduler based on a delay-counting function and the equalisation property of the underlying WFQ discipline could be developed.

[12]assuming that each ACK packet acknowledges one or two segments.
[13]the kernel-level APIs are not clear at the time of this writing.
[14]a suitable cost function.

11.3 Evolution of TCP

11.3.1 Achievements

The contribution done in this part of my work is the evaluation and the analysis of the behaviour of TCP in wireless scenarios relevant for WLAN-Internet integration. Moreover the investigation of these scenarios leads to the conclusion, that classical (Reno) TCP may in general be a deficitary algorithm with respect to the short-time fairness as well as to long-term fairness (in the case of WLAN) under certain circumstances. The intended congestion avoidance of deployed TCP algorithms does not always successfully avoid network congestion, e.g. in a few-users scenario as imposed by a wireless-wired integration situation. The core of the fairness problem[15] is the aggressive window probing of Reno-like TCP, which must overflow the bottleneck link queue first, in order to estimate the available capacity along its path. Improvements of the widespread Reno-like TCP algorithm like Vegas-TCP as studied through NS simulations don't seem to substantially improve the perceived performance with respect to throughput fairness and congestion avoidance (that is low end-to-end delay).

Based on those findings an idea for implementing a differential type of congestion control is developed and investigated in a preliminary TCP-D prototype for the NS2 network simulator. The brief analysis of the TCP-D prototype shows that the idea can work and can further improve the end-to-end performance with respect to short-term but also long-term throughput fairness. Moreover an improved congestion avoidance algorithm like TCP-D can help to alleviate the permanent network congestion level. The key idea of TCP-D is based on an analogy to a harmonic oscillator and deploys a finer-grained control of the congestion window size than in Reno-TCP. The rationale for developing TCP-D is based on the observation that classical congestion control algorithms[16] are blind to the absolute level of network congestion and rely solely on the induction of heavy congestion[17], in order to establish the right point of operation.

11.3.2 Perspectives

A follow-up research work must be conducted and should investigate the mathematical model as proposed for TCP-D in more detail. Especially a more formal derivation of the control equation based on the evidence of the existence of a harmonic-type "potential" in the proximity of the "optimal" operation point of a TCP source should be presented.

There are more pending issues to be resolved in the practical implementation of TCP-D. An improved prototype of TCP-D must utilise higher accuracy for the computation of the bandwidth derivative and could for example use an extrapolation technique to obtain a continuous estimation of the ξ value and thus become less dependent of the size[18] of the sampling interval k. There exist numerous tech-

[15]not to be confused with the download-upload asymmetry problem in 802.11.
[16]with exception of Vegas-TCP and ECN.
[17]in absence of RED or other active management schemes.
[18]that is the number of points used

niques for computing derivatives or integrals, if the function value is available only at a discrete and limited number of points[19] in the definition interval. One of the techniques is polynomial approximation which I believe is the next improvement to be tested in the TCP-D prototype. The idea is to approximate the discrete function as imposed by the transferred data volume over time by a suitable[20] polynomial and calculate the derivative of the approximated function as the value of the derivative of the polynomial.

Unfortunately high-precision mathematics is a task, which may encounter practical limits in a real-world implementation, where computing power is usually a limited ressource - unlike in a simulated NS2 environment, where the real computation time necessary for processing of each packet is not significant. The prevalent domination of Reno-type TCP in today Internet can be contributed to its really low algorithmic complexity, its simple structure and the relative ease of implementation[21].

Improving the TCP congestion avoidance and the cold-start behaviour inevitably leads to increased algorithmic complexity and requires higher computing power in the end hosts. This observation can impose a limit on the further development in this area. It is worth to note that for example with techniques like Gigabit or even 10 Gigabit Ethernet already the classical Reno-TCP requires a substantial amount of computing power to be spent on checksumming and on memory copying operations and therefore imposes a natural limit for practical usability[22] of those networking technologies in consumer PC hardware.

This technological limitation may disappear with advances in hardware architectures (e.g. improvements in bus systems like the PCIe bus system or more hardware support for TCP (acceleration) or silicon TCP/IP stacks). Since the average utilisation scenario of the Internet for the majority of end-users means "download", the effort needed to perform improved and more complex congestion control is going to concentrate at those server machines providing the consumed content. Therefore any further acceptance and deployment of improved congestion control algorithms may be limited. One possible way out of this dilemma would be the investigation of a receiver-side congestion control, in order to distribute the computation power onto the client machines.

Second open TCP-D issue is its fair sharing of bandwidth with other TCP algorithms, however as explained in the analysis of TCP-Reno, the behaviour of Reno should not be taken as reference. There is an obvious relation between fairness of collision avoidance algorithms as analysed in the first part of this work and analogous fairness issue in TCP-like congestion avoidance as analysed in the second part. The two problems are not equal but are very related and in some sense "inverses" of each other: while a collision avoidance scheme is trying (by increasing the contention window size, that is by lowering the channel pressure) to find a working point where the collision rates are sufficiently low, a congestion avoidance scheme is trying to

[19]numerical mathematic.

[20]that is of a suitable grade.

[21]However it should be mentioned, that despite this fact the TCP/IP stack is one of the most complex subsystems of an operating system e.g. like the Linux kernel.

[22]if the speed should be used.

find a working point where the resulting throughput is sufficiently high (that is by increasing the pressure) but not too harmfull to the transmission path.

The correlation analysis approach as presented and applied to the collision avoidance problem may be similarly extended to quantify the deviation of a congestion avoidance scheme from a reference system, which still must be found in case of TCP in a future research work. The analogy between the two problems is obvious: while in 802.11 a station which has performed the exponential backoff procedure cuts the size of its contention window to the minimum value of CW_min, in TCP a station[23] cuts the congestion window to half after experiencing heavy congestion. Therefore the short-term fairness deficiency of Reno-TCP can be quantified by studying the self-correlation index of the sending station. This is subject to further research.

Another pending issue in the area of improving the performance of TCP is the introduction of probabilistic elements into TCP's congestion avoidance. The deterministic behaviour of classical TCP in some of its possible operation regimes has been identified as the source of degraded end-to-end performance (window synchronisation, shut-off connections, etc.). Introducing randomness into TCP has been found [111] to improve the performance in these cases, however the randomness was considered in literature just as processing time randomness, not as algorithmic randomness, e.g. as widely used for the 802.11 collision avoidance scheme. Therefore there is an open research field for augmenting the congestion avoidance but also the cold-start behaviour of TCP by random elements.

[23]**the TCP controlling a particular socket.**

Bibliography

[1] O. Abuamsha and N. Pekergin, "*Comparison of Fair Queuing Algorithms with a Stochastic Approach*", in Proc. MASCOTS, 1998.

[2] A. Aggarwal, S. Savage and T. Anderson, "*Understanding the Performance of TCP Pacing*", in Proc. INFOCOM, 2000.

[3] O. Ait-Hellal and E. Altman, "*Analysis of TCP Vegas and TCP Reno*", Journal of Telecommunication Systems, volume 15, no. 3-4, 2000.

[4] M. Allman, "*On the Generation and Use of TCP Acknowledgments*", ACM SIGCOMM Comput. Commun. Rev., volume 28, no. 5, 1998.

[5] M. Allman, "*TCP Byte Counting Refinements*", ACM SIGCOMM Comput. Commun. Rev., volume 29, no. 3, 1999.

[6] M. Allman and A. Falk, "*On the Effective Evaluation of TCP*", ACM SIGCOMM Comput. Commun. Rev., volume 29, no. 5, 1999.

[7] T. Anker, R. Cohen, D. Dolev and Y. Singer, "*PrFQ: Probabilistic Fair Queuing*", Technical Report CS-2000-30, Institute of Computer Science, The Hebrew University of Jerusalem, Israel, 2000.

[8] Apache Software Foundation, "*Apache HTTP Server Version 1.3 API notes*", 2006.
URL http://httpd.apache.org/docs/1.3/misc/API.html

[9] Apache Software Foundation, "*The Apache HTTP Server Project*", 2006.
URL http://httpd.apache.org/

[10] J. Aweya, M. Ouellette and D. Y. Montuno, "*A Self-Regulating TCP Acknowledgment (ACK) Pacing Scheme*", Int. J. Netw. Manag., volume 12, no. 3, 2002.

[11] A. Bakre and B. R. Badrinath, "*I-TCP: Indirect TCP for Mobile Hosts*", in Proc. 15th International Conference on Distributed Computing Systems ICDCS, 1995.

[12] A. Balachandran, G. Voelker, P. Bahl and P. Rangan, "*Characterizing User Behavior and Network Performance in a Public Wireless LAN*", in Proc. ACM SIGMETRICS, 2002.

[13] H. Balakrishnan, V. N. Padmanabhan, S. Seshan and R. H. Katz, "*A Comparison of Mechanisms for Improving TCP Performance over Wireless Links*", IEEE/ACM Transactions on Networking, volume 5, no. 6, 1997.

[14] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm and R. H. Katz, "*TCP Behavior of a Busy Internet Server: Analysis and Improvements*", in Proc. IEEE INFOCOM, 1998.

[15] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "*Efficient TCP over Networks with Wireless Links*", in Proc. 1st ACM International Conference on Mobile Computing and Networking, 1995.

[16] H. Balakrishnan, S. Seshan, E. Amir and R. H. Katz, "*Improving TCP/IP Performance over Wireless Networks*", in Proc. 1st ACM International Conference on Mobile Computing and Networking, 1995.

[17] H. Balakrishnan, S. Seshan and R. H. Katz, "*Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks*", Journal of ACM Wireless Networks, 1995.

[18] R. Battiti and B. Li, "*Supporting Service Differentiation With Enhancements Of The IEEE 802.11 MAC Protocol: Models And Analysis*", Technical Report University of Trento, Department of Information and Communication Technology, Italy, 2003.

[19] J. C. Bennett and H. Zhang, "*Hierarchical Packet Fair Queuing Algorithms*", in Proc. ACM SIGCOMM, 1996.

[20] J. C. Bennett and H. Zhang, "*WF^2Q: Worst-Case Fair Weighted Fair Queueing*", IEEE INFOCOM, 1996.

[21] G. Berger-Sabbatel, A. Duda, O. Gaudoin, M. Heusse and F. Rousseau, "*Fairness and its Impact on Delay in 802.11 Networks*", in Proc. IEEE GLOBECOM, 2004.

[22] G. Bianchi, "*Performance Analysis of the IEEE Distributed Coordination Function*", Journal on Selected Areas in Communications, volume 18, no. 3, 2000.

[23] Bluetooth SIG Inc., "*The Official Bluetooth Website*", 2005.
URL http://www.bluetooth.com

[24] T. Bonald, "*Comparison of TCP Reno and TCP Vegas via Fluid Approximation*", Technical Report INRIA RR-3563.
URL http://www-sop.inria.fr/rapports/sophia/RR-3563.html

[25] L. S. Brakmo, S. W. O'Malley and L. L. Peterson, "*TCP Vegas: New Techniques for Congestion Detection and Avoidance*", in Proc. ACM SIGCOMM, 1994.

[26] L. S. Brakmo and L. L. Peterson, "*TCP Vegas: End to End Congestion Avoidance on a Global Internet*", IEEE Journal on Selected Areas in Communications, volume 13, no. 8, 1995.

[27] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu and H. Yu, "*Advances in Network Simulation*", IEEE Computer, volume 33, no. 5, 2000.
URL http://www.isi.edu/~johnh/PAPERS/Bajaj99a.html

[28] R. Cacares and L. Iftode, "*Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments*", IEEE Journal on Selected Areas in Communications, volume 13, no. 5, 1995.

[29] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi and R. Wang, "*TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks*", Wireless Networks, volume 8, no. 5, 2002.

[30] CERN, European Organization for Nuclear Research, "*The LHC Computing Grid Project*", CH-1211, Genève 23, Switzerland, 2005.
URL http://lcg.web.cern.ch/lcg/

[31] J. Choi, J. Yoo, S. Choi and C. Kim, "*EBA: An Enhancement of the IEEE 802.11 DCF via Distributed Reservation*", IEEE Transactions on Mobile Computing, volume 4, no. 4, 2005.

[32] S. Choi, J. del Prado, S. Shankar and S. Mangold, "*IEEE 802.11e Contention Based Channel Access (EDCF) Performance Evaluation*", in Proc. IEEE ICC, 2003.

[33] K. Claffy, G. J. Miller and K. Thompson, "*The nature of the beast: recent traffic measurements from an Internet backbone*", in Proc. ISOC INET Conference, 1998.

[34] K. A. L. Coar and D. R. T. Robinson, "*RFC-DRAFT: The WWW Common Gateway Interface Version 1.1*", 1999.
URL http://ietfreport.isoc.org/all-ids/draft-coar-cgi-v11-03.txt

[35] R. Davies, "*Hardware Random Number Generators*", 15th Australian Statistics Conference, 2000.

[36] R. de Oliveira and P. R. Guardieiro, "*A Comparative Study Of TCP Reno And Vegas In A Differentiated Services Network*", in Proc. 3rd Conference on Telecommunications (ConfTele), 2001.

[37] A. Demers, S. Keshav and S. Shenker, "*Analysis and Simulation of a Fair Queueing Algorithm*", in Proc. ACM SIGCOMM, 1989.

[38] A. Detti, E. Graziosi and V. Minichiello, "*TCP Fairness Issues in IEEE 802.11 Based Access Networks*", submitted paper, 2005.
URL http://www.eln.uniroma2.it/Stefano_Salsano/papers/salsano-tcp-fair-wlan.pdf

[39] Digital Equipment Corporation, Intel Corporation, Xerox Corporation, "*The Ethernet - A Local Area Network, Version 1.0*", 1980.
URL http://grouper.ieee.org/groups/802/3/index.html

[40] D. R. Dooly, S. A. Goldmann and S. D. Scott, "*TCP Dynamic Acknowledgement Delay: Theory and Practice*", 13th Annual ACM Symposium on Theory of Computing, 1998.

[41] D. Eckhardt and P. Steenkiste, "*Effort-Limited Fair (ELF) Scheduling for Wireless Networks*", in Proc. IEEE INFOCOM, 2000.

[42] S. Floyd, "*Issues of TCP with SACK*", Technical report, Lawrence Berkeley National Laboratory, 1996.
URL http://www.icir.org/floyd/papers/issues_sa.ps.Z

[43] S. Floyd, "*On the Evolution of End-to-end Congestion Control in the Internet. An Idiosyncratic View*", Technical report, 1999.
URL http://www.icir.org/floyd/talks/sf-evolution-Apr99.pdf

[44] S. Floyd, R. Gummadi and S. Shenker, "*Adaptive RED: An Algorithm for Increasing the Robustness of RED*", Technical report, 2001.
URL http://www.icir.org/floyd/papers/adaptiveRed.pdf

[45] S. Floyd and T. Henderson, "*RFC 2582: New Reno Modification to TCP's Fast Recovery*", 1999.
URL http://www.faqs.org/rfcs/rfc2582.html

[46] S. Floyd and V. Jacobson, "*Random Early Detection Gateways for Congestion Avoidance*", IEEE/ACM Transactions on Networking, volume 1, no. 4, 1993.

[47] C. P. Fu, "*TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks*", Ph.D. thesis, Chinese University of Hong Kong, 2001.

[48] S. J. Golestani, "*A Self-Clocked Fair Queueing Scheme for Broadband Applications*", in Proc. IEEE INFOCOM, 1994.

[49] P. Goyal, H. M. Vin and H. Cheng, "*Start-Time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks*", IEEE/ACM Transactions on Networking, volume 26, no. 4, 1997.

[50] L. A. Grieco and S. Mascolo, "*TCP Westwood and Easy RED to Improve Fairness in High-Speed Networks*", in Proc. 7th IFIP/IEEE International Workshop on Protocols for High Speed Networks, 2002.

[51] L. A. Grieco and S. Mascolo, "*Performance Evaluation and Comparison of Westwood+, New Reno and Vegas TCP Congestion Control*", ACM SIGCOMM Comput. Commun. Rev., volume 34, no. 2, 2004.

[52] R. L. Grossman, Y. Gu, Xinwei, A. Antony, J. Blom, F. Dijkstra and C. de Laat, "*Experimental Studies Using Application Layer Protocols Based Upon UDP to Support Applications Requiring Requiring Very High Volume Data Flows*", Technical report, 2004.
URL http://staff.science.uva.nl/~fdijkstr/publications/journal-030.pdf

[53] Y. Gu and R. Grossman, "*SABUL: A Transport Protocol for Grid Computing*", J-GRID-COMP, volume 1, no. 4, 2003.

[54] Y. Gu and R. Grossman, "*UDT: A Transport Protocol for Data Intensive Applications*", Technical report, Laboratory for Advanced Computing, University of Illinois, Chicago, 2004.
URL http://www.rnp.br/ietf/internet-drafts/draft-gg-udt-00.txt

[55] P. Hadam, "*Transports Nouvelle Génération dans les Réseaux à Très Haut Débit*", Ph.D. thesis, INPG, 2005.

[56] M. Handley, S. Floyd, J. Pahdye and J. Widmer, "*RFC 3448: TCP Friendly Rate Control (TFRC): Protocol Specification*", 2003.
URL http://www.faqs.org/rfcs/rfc3448.html

[57] U. Hengartner, J. Bolliger and T. Gross, "*TCP Vegas Revisited*", in Proc. INFOCOM, 2000.

[58] M. Heusse, F. Rousseau, G. Berger-Sabbatel and A. Duda, "*Performance Anomaly of 802.11b*", in Proc. IEEE INFOCOM, 2003.

[59] M. Heusse, F. Rousseau, G. Berger-Sabbatel and A. Duda, "*Short-Term Fairness of 802.11 Networks with Several Hosts*", in Proc. 6th IFIP IEEE International Conference on Mobile and Wireless Communication Networks (MWCN), 2004.

[60] M. Heusse, F. Rousseau, R. Guillier and A. Duda, "*Idle Sense : An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs*", in Proc. GLOBECOM, 2005.

[61] M. Heusse, P. Starzetz, F. Rousseau, G. Berger-Sabbatel and A. Duda, "*Bandwith Allocation for DiffServ based Quality of Service over 802.11b*", in Proc. IEEE GLOBECOM, 2003.

[62] M. Heusse, P. Starzetz, F. Rousseau, G. Berger-Sabbatel and A. Duda, "*Scheduling Time-sensitive Traffic on 802.11 Wireless LANs*", in Proc. 4th COST 263 International Workshop on Quality of Future Internet Services (QoFIS), 2003.

[63] G. Holland and N. Vaidya, "*Analysis of TCP performance over Mobile Ad-Hoc Networks*", Wireless Networks, volume 8, no. 2/3, 2002.

[64] W. Hörmann and J. Leydold, "*Continuous Random Variate Generation by Fast Numerical Inversion*", j-TOMACS, volume 13, no. 4, 2003.
URL http://statistik.wu-wien.ac.at/unuran/

[65] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. B. Schroeder, J. Spaans and P. Larroy, "*Linux Advanced Routing and Traffic Control*", 2004.
URL http://lartc.org/

[66] IEEE, "*ANSI/IEEE Std. 1999 Edition Part 11, Wireless LAN Medium Access Controll (MAC) and Physical Layer (PHY specification)*", Technical report, 1999.
URL http://standards.ieee.org/catalog/olis/lanman.html

[67] IEEE, "*Draft Supplement to Standard for Telecommunications and Information Exchange Between Systems -LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: MAC Enhancements for Quality of Service (QoS), IEEE 802.11e/D2.0, IEEE 802.11e WG.*", Technical report, 2001.

[68] Information Sciences Institute University of Southern California, "*RFC 793: TRANSMISSION CONTROL PROTOCOL*", 1981.
URL http://www.faqs.org/rfcs/rfc793.html

[69] Intersil, "*Prism Driver Programmers Manual*", available from Intersil after signing a NDA, 2002.

[70] @IRS Project, "*Integrated Architecture for Networks and Services*", 2001.
URL http://www-rp.lip6.fr/airs/

[71] V. Jacobson, R. Braden and D. Borman, "*RFC 1323: TCP Extensions for High Performance*", 1992.
URL http://www.faqs.org/rfcs/rfc1323.html

[72] V. Jacobson and R. T. Braden, "*RFC 1072: TCP Extensions for Long Delay Paths*", 1988.
URL http://www.faqs.org/rfcs/rfc1072.html

[73] V. Jacobson, C. Leres and S. McCanne, "*TCPDump Public Repository*", University of California Berkeley CA, 2005.
URL http://www.tcpdump.org/

[74] R. Jain, D. Chiu and W. Hawe, "*A Quantitative Measure of Fairness and Discrimination for Resource Allocation in SharedComputer Systems*", Technical report, 1984.

[75] Jeff Nathan, <jeff@snort.org>, "*Nemesis: Universal Packet Generation Suite*", 2005.
URL http://nemesis.sourceforge.net/

[76] A. Kammerman, "*Spread Spectrum Schemes for Microvawe-Frequency WLANs*", Microwave Journal, volume 40, no. 2, 1997.

[77] S. S. Kanhere, H. Sethu and A. B. Parekh, "*Fair and Efficient Packet Scheduling Using Elastic Round Robin*", IEEE Transactions on Parallel and Distributed Systems, volume 13, no. 3, 2002.

[78] F. Kelly, "*Mathematical Modeling of the Internet*", in Proc. 4th International Congress on Industrial and Applied Mathematics, 1999.

[79] T. Kelly, "*Scalable TCP: Improving Performance in Highspeed Wide Area Networks*", ACM SIGCOMM Comput. Commun. Rev., volume 33, no. 2, 2003.

[80] S. Keshav, "*A Control-Theoretic Approach to Flow Control*", in Proc. ACM SIGCOMM, 1991.

[81] S. Khurana, A. Kahol, S. K. S. Gupta and P. K. Srimani, "*Performance Evaluation of Distributed Co-Ordination Function for IEEE 802.11 Wireless LAN Protocol in Presence of Mobile and Hidden Terminals*", in Proc. MASCOTS, 1999.

[82] A. Koepsel, J.-P. Ebert and A. Wolisz, "*A Performance Evaluation of Point and Distributed Coordination Function of an IEEE 802.11 WLAN in the Presence of Real Time Requirements*", in Proc. 7th Int. Workshop on Mobile Multimedia Communication, 2000.

[83] C. E. Koksal, H. Kassab and H. Balakrishnan, "*An Analysis of Short-Term Fairness in Wireless Media Access Protocols*", in Proc. ACM SIGMETRICS, 2000.

[84] S. H. Kung, "*Use of TCP Decoupling in Improving TCP Performance over Wireless Networks*", Wireless Networks, volume 7, no. 3, 2001.

[85] S. Kuppa and R. Prakash, "*Service Differentiation Mechanisms for IEEE 802.11-based Wireless Networks*", IEEE WCNC, 2004.

[86] J. Leydold and W. Hörmann, "*UNURAN - A library for Universal Non-Uniform Random Number Generators*", Technical report, Institut für Statistik, WU Wien, A-1090 Wien, Austria, 2000.
URL ftp://statistik.wu-wien.ac.at/src/unuran/

[87] C. Lim and C. Choi, "*TDM-based Coordination Function in WLAN for High Throughput*", in Proc. IEEE GLOBECOM, 2004.

[88] A. Lindgren, O. Schelén and A. Almquist, "*Evaluation of Quality of Service Schemes for IEEE 802.11 Wireless LANs*", in Proc. 26th Annual IEEE Conference on Local Computer Networks, 2001.

[89] Linksys, "*Division of Cisco Systems Inc.*", 2005.
URL http://www.linksys.com

[90] C. Liu and R. Jain, "*Improving Explicit Congestion Notification with the Mark-Front Strategy*", Comput. Networks, volume 35, no. 2-3, 2001.

[91] J. A. G. Macias, "*Mobile Communication Architecture with Quality of Service*", Ph.D. thesis, INPG, 2002.

[92] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "*RFC 2018: TCP Selective Acknowledgement Options*", 1996.
URL http://www.faqs.org/rfcs/rfc2018.html

[93] S. McCanne and S. Floyd, "*The NS2 Network Simulator, Version 2.28*", 2005.
URL http://www.isi.edu/nsnam/ns/

[94] P. McKenney, "*Stochastic Fairness Queuing*", IEEE INFOCOM, 1990.

[95] J. Mo, R. J. La, V. Anantharam and J. C. Walrand, "*Analysis and Comparison of TCP Reno and Vegas*", in Proc. IEEE INFOCOM, 1999.

[96] J. Mogul, "*Observing TCP Dynamics in Real Networks*", in Proc. ACM SIGCOMM, 1992.

[97] R. Neelamani and A. Saxena, "*TCP over Wireless*", Technical Report Elec 537 Course Project, University, Houston, TX 77005.
URL http://www-dsp.rice.edu/~neelsh/publications/TCP.ps.gz

[98] Netfilter core team, H. Welte, J. Kadlecsik, M. Josefsson and P. McHardy, "*Netfilter, Firewalling, NAT and Packet Mangling for Linux*", 2005.
URL http://www.netfilter.org/

[99] T. S. E. Ng, I. Stoica and H. Zhang, "*Packet Fair Queueing Algorithms for Wireless Networks with Location-Dependent Errors*", in Proc. INFOCOM, 1998.

[100] Q. Ni, I. Aad, C. Barakat and T. Turletti, "*Modeling and Analysis of Slow CW Decrease for IEEE 802.11 WLAN*", in Proc. IEEE PIMRC, 2003.

[101] Q. Ni, L. Romdhani, T. Turletti and I. Aad, "*QoS Issues and Enhancements for IEEE 802.11 Wireless LAN*", Technical Report INRIA RR-4612, 2002. URL http://icapeople.epfl.ch/iaad/publ/RR-4612.pdf

[102] L. Ong and J. Yoakum, "*RFC 3286: An Introduction to the Stream Control Transmission Protocol (SCTP)*", 2002. URL http://www.faqs.org/rfcs/rfc3286.html

[103] J. Pahdye and S. Floyd, "*On Inferring TCP Behavior*", in Proc. ACM SIGCOMM, 2001.

[104] A. K. Parekh and R. G. Gallager, "*A Generalized Processor Sharing Approach to Flow Control in Integrated Services Network: The Single Node Case*", IEEE/ACM Transactions on Networks, volume 1, no. 3, 1993.

[105] C. Parsa and J. Garcia-Luna-Aceves, "*TULIP: A Link-Level Protocol for Improving TCP over Wireless Links*", in Proc. IEEE WCNC, 1999.

[106] C. Parsa and J. J. Garcia-Luna-Aceves, "*Improving TCP Performance over Wireless Networks at the Link Layer*", Mobile Networks and Applications, volume 5, no. 1, 2000.

[107] V. Paxon, M. Handley and C. Kreibich, "*Network Intrusion Detection: Evasion, Traffic Normalization and End-to-End Protocol Semantics*", in Proc. USENIX Security Symposium, 2001.

[108] K. Pentikousis and H. Badr, "*An Evaluation of TCP with Explicit Congestion Notification*", Annals of Telecommunications, volume 59, no. 1-2, 2004.

[109] K. Pentikousis, H. Badr and B. Kharmah, "*TCP with ECN: Performance Gains for Large Transfers*", Technical Report SBCS-TR-2001/01, Department of Computer Science, SUNY Stony Brook, 2001.

[110] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt and P. Sinha, "*Understanding TCP Fairness over Wireless LAN*", in Proc. IEEE INFOCOM, 2003.

[111] L. Qiu, Y. Zhang and S. Keshav, "*Understanding the Performance of Many TCP Flows*", Comput. Networks, volume 37, no. 3-4, 2001.

[112] M. Radimirsch and V. Vollmer, "*HIPERLAN Type 2 Standardisation - An Overview*", in Proc. European Wireless Conference, 1999.

[113] K. K. Ramakrishnan and S. Floyd, "*RFC 2481: A Proposal to Add Explicit Congestion Notification to IP*", 1999. URL http://www.faqs.org/rfcs/rfc2481.html

[114] P. Ramanathan and P. Agrawal, "*Adapting Packet Fair Queueing Algorithms to Wireless Networks*", in Proc. 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1998.

[115] B. Rathke, M. Schlager and A. Wolisz, "*Systematic Measurement of TCP Performance over Wireless LANs*", Technical Report TKN-01BR98, Telecommunication Networks Group, Technical University Berlin, 1998.

[116] RealNetworks Inc., "*Real Player version 10*", 2601 Elliott Avenue, Seattle, WA 98121 U.S.A., 2005.
URL http://www.real.com/

[117] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "*RFC 3550: RTP, A Transport Protocol for Real-Time Applications*", 2003.
URL http://www.faqs.org/rfcs/rfc3550.html

[118] Security Portal for Information System Security Professionals, "*Denial of Service Attacks - DDOS, SMURF, FRAGGLE, TRINOO*", 2006.
URL http://www.infosyssec.org/infosyssec/security/secdos1.html

[119] V. A. Siris and M. Kavouridou, "*Achieving Service Differentiation and High Utilization in IEEE 802.11*", Technical Report 322, 2003.
URL http://www.ics.forth.gr/netlab/publications/2003.TR322.sd_802.11.pdf

[120] J. L. Sobrinho and A. S. Krishnakumar, "*Quality-of-Service in Ad-Hoc Carrier Sense Multiple Access Networks*", IEEE Journal on Selected Areas in Communications, volume 17, no. 8, 1999.

[121] P. Starzetz, "*Ambiguities in TCP/IP - Firewall Bypassing*", Bugtraq Article, 2002.
URL http://seclists.org/lists/bugtraq/2002/Oct/0274.html

[122] W. Stevens, "*RFC 2001: TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*", 1997.
URL http://www.faqs.org/rfcs/rfc2001.html

[123] R. Stewart, Q. Xie and et al., "*RFC 2960: Stream Control Transmission Protocol*", 2000.
URL http://www.faqs.org/rfcs/rfc2960.html

[124] Y. Tamura, Y. Tobe and H. Tokuda, "*EFR: A Retransmit Scheme for TCP in Wireless LANs*", in Proc. 23rd Annual IEEE Conference on Local Computer Networks, 1998.

[125] L. Torvalds, "*The Linux Kernel Archives*", 2006.
URL http://www.kernel.org

[126] V. Tsaoussidis and I. Matta, "*Open issues on TCP for mobile computing*", Technical Report 013, 2001.

[127] N. H. Vaidya, P. Bahl and S. Gupta, "*Distributed Fair Scheduling in a Wireless LAN*", in Proc. MOBICOM, 2000.

[128] S. Vangala and M. Labrador, "*Performance of TCP over Wireless Networks with the Snoop Protocol*", in Proc. IEEE LCN, 2002.

[129] W3C, "*HTTP - Hypertext Transfer Protocol*", 2005.
URL http://www.w3.org/Protocols/

[130] R. Wang, G. Pau, K. Yamada, M. Sanadidi and M. Gerla, "*TCP Startup Performance in Large Bandwidth Delay Networks*", IEEE INFOCOM, 2004.

[131] E. Weigle and W. C. Feng, "*A Case for TCP Vegas in High-Performance Computational Grids*", in Proc. 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01), 2001.

[132] J. Weinmiller, M. Schlaeger, A. Festag and A. Wolisz, "*Performance Study of Access Control in Wireless LANs: IEEE 802.11 DFWMAC and ETSI RES 10 Hiperlan*", Mobile Networks and Applications, volume 2, no. 1, 1997.

[133] J. M. Winett, "*RFC 147: The Definition of a Socket*", 1971.
URL http://www.faqs.org/rfcs/rfc147.html

[134] H. T. Wu and S. D. Cheng, "*DCF+: an Enhancement for Reliable Transport Protocol over WLAN*", J. Comput. Sci. Technol., volume 18, no. 2, 2003.

[135] K. Xu, Y. Tian and N. Ansari, "*TCP-Jersey for Wireless IP Communications*", IEEE JSAC, volume 22, no. 4, 2004.

[136] G. Xylomenos and G. C. Polyzos, "*TCP and UDP Performance over a Wireless LAN*", in Proc. IEEE INFOCOM, 1999.

[137] G. Ye, T. Saadawi and M. Lee, "*SCTP Congestion Control Performance in Wireless Multihop Networks*", in Proc. MILCOM, 2002.

[138] C. Yeh and H. Zhou, "*A New Class of Collision-Free MAC Protocols for Ad-Hoc Wireless Networks*", in Proc. International Conference Advances in Infrastructure for e-Business, e-Education, eScience, and e-Medicine on the Internet, 2002.

[139] M. Youssef, A. Vasan and R. Miller, "*Specification and Analysis of the DCF and PCF Protocols in the 802.11 Standard Using Systems of Communicating Machines*", in Proc. IEEE ICNP, 2002.

[140] V. C. Zandy and B. P. Miller, "*Reliable Network Connections*", in Proc. MOBICOM, 2002.

Thanks

My thanks goes in first place to Mr. Andrzej Duda for accepting my candidature for a PhD student for his Drakkar research group at IMAG-LSR and helping me together with Mr. Sacha Krakowiak to pass through the initial registration and enrolment. Additionaly I also want to thank Mr. Duda for his liberal and non-pressurisng supervision of the PhD work which permitted me to go into the desired direction and not to stick to only one thing.

I want to thank Mme. Pascale Primet Vicat-Blanc and Mr. Laurent Toutain for providing their expert advice and evaluating this thesis, which has certainly required a lot of their spare and valuable time. Simultaneously I want to thank the other members of the doctoral jury namely Mr. Jacques Chassin de Kergommeaux and Mr. Franck Rousseau for agreeing to participate in the PhD defense.

Special thanks goes also to Mr. Martin Heusse for useful and interesting discussions on the subject of wireless access algorithms and for reading the critical sections of my work, giving me valuable feedback which permitted me to improve the quality of this manuscript.

Great thanks must also go to all the students who helped me with implementation of some of those small ideas that I had during my roughly 3.5 year period at LSR, namely: Krzysztof Mazur and Piotr Nowakowski for helping me with basic NS2 stuff, Michal Kosek for providing aid in working on Linux VFQ, Tomasz Kaszta for similar aid in VFQ for NS2, and also to the other members of the Drakkar team: Gilles Berger-Sabbatel and Dominique Decouchant.

I'm also very grateful to all the people that I've met at LSR showing enough patiente with a stranger like me, namely my initial colleagues Stephane Lo-Presti (for providing me also useful insights into French culture and the role of a PhD student), Hoa-Binh Nguyen (for being a special friend all of the time), PawełHadam, Vincent Untz and Cristina Pop for their patience with my sometimes strange habitudes, Laurentiu-Sorin Paun, Justinian Oprescu, Leyla Toumi, Phuong-Hoang Nguyen, and Sonia Mendoza for being my special friend during the last year of my PhD time.

Abstract

This thesis has a threefold contribution. The first part provides a statistical approach at analysing CSMA/CA type channel access algorithms with the focus on the IEEE 802.11 WLAN standard and delivers novel insights into the structure of the IEEE 802.11 CSMA/CA access algorithm. The source of the short-term fairness in CSMA/CA type protocols is identified and the classical Jain's fairness widely index used to quantify short-time fairness is complemented by channel autocorrelation function.

The second part of this work investigates a WLAN/LAN integration scenario in the context of the end-to-end performance of todays Internet protocols with focus on the classical Transmission Control Protocol (TCP). Performance issues of standard TCP in WLAN/LAN integration scenario are identified and an active wireless queue management solution for TCP based on generalised fair cost scheduling called VFQ is presented.

In the third and last part finally, the behaviour of the TCP protocol in simple wireless intergation scenarios is analysed and a number of undesirable properties recognised and used to design an improvement of the classical TCP congestion control with focus on the short-time fairness.