Budapest University of Technology and Economics

Design and Analysis of New Control Mechanisms to Support Data Traffic in Mobile Packet Networks

György Miklós

High Speed Networks Laboratory
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Ph. D. Dissertation

Advisor:

Dr. Sándor Molnár
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics

Budapest, 2004

Budapesti Műszaki és Gazdaságtudományi Egyetem

Mobil csomagkapcsolt hálozatok adatforgalmazását támogató új szabalyozó mechanizmusok tervezése és elemzése

Miklós György

Nagysebessegű Hálózatok Laboratóriuma
Távközlési es Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Ph. D. disszertáció

Tudományos vezető:

Dr. Molnár Sándor
Távközlési es Médiainformatikai Tanszék
Budapesti Műszaki és Gazdaságtudományi Egyetem

Budapest, 2004

Contents

A bírálatok és a tézisfüzet a BME Villamosmérnöki és Informatikai Kar Dékáni Hivatalában megtekinthetők.

Acknowledgments

I am grateful to many people who have directly or indirectly helped my work. First of all I am thankful to my supervisor, Dr. Sándor Molnár, who has always been very motivated and motivating about my scientific work. He has encouraged me to look at the heart of problems and work on the fundamental theoretical questions that answer practical issues. While he takes science very seriously, he is also a bright personality who has been a pleasure to work with. I have been very fortunate to have him provide support all through my Ph.D. studies.

I thank the work of Dr. Tamás Henk, the head of High Speed Networks Laboratory, who has worked to provide good research environment. I am also grateful to Miklós Boda, head of the Research Lab at Ericsson Hungary, who has not only provided a lot of financial support for my studies and provided the best means of education, but also gave me and my fellows a very strong sense of focus and determination in our work. In many occasions he believed in us more than we believed in ourselves, and this has been very encouraging. Together with Hans Eriksson, head of Traffic Analysis and Network Performance Lab at Ericsson Hungary, they have always encouraged me to take the time and energy for the dissertation.

During my studies I have had the opportunity to take part in the research, development and standardization work of the HIPERLAN/2 wireless LAN standard at Ericsson Research. In this project I have learnt not only how to address difficult technical questions, but also how to work efficiently in a team. The strong commitment of the project members has impressed me. I am thankful to Göran Malmgren, the leader of this effort, for his guidance and trust in me.

I have been lucky to work together with the bright people at Traffic Analysis and Network Performance Lab. My work in the Limes project has been very exciting and also contributed to the dissertation a lot. I have enjoyed the work with my colleagues: András Rácz, Zoltán Turányi, András Valkó, Ferenc Kubinszky, Márk Félegyházi, Miklós Aurél Rónai and others. As the leader of the activites, András Valkó has fostered an atmosphere where we can feel free to work on our own and at the same time work towards our common goals.

In my work I have often used simulation as a tool in research. For the simulations I have used the PLASMA simulation environment most of the time. In doing so, I have used the outcome of the co-operation between Ericsson and the High Speed Networks Laboratory at the Budapest University of Technology and Economics, Department of Telecommunications and Media Informatics. While this tool needs some initial work investment to learn in the beginning, I have found that it is well worth the price and simulation results have provided numerous insights and have improved the solutions presented in the dissertation a lot. In the simulation work, I worked together with Miklós Aurél Rónai, David Gabbitas and David Shay.

My first projects in communication networks were completed at the Lund Institute of Technology, Department of Communication Systems, under Johan Karlsson. Under his tolerant guidance I could enjoy my work as well as my time in Sweden.

Together with my colleagues I had the opportunity to work together with Arata Koike of NTT, Japan. I had learnt a lot from this co-operation which has tested our skills and working capacity. I thank Péter Tatai for this opportunity as well as for his guidance all along.

I also thank my family that has supported me in everything I do.

List of Abbreviations

ACH	Access feedback CHannel
AP	Access Point
ATM	Asynchronous Transfer Mode
ARQ	Automatic Repeat reQuest
BBP	Batch Bernoulli Process
BCH	Broadcast CHannel
BER	Bit Error Rate
BRAN	Broadband Radio Access Networks
C-PDU	Control Protocol Data Unit
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DFS	Dynamic Frequency Selection
DLC	Data Link Control
DL	DownLink
ETSI	European Telecommunication Standards Institute
FCH	Frame CHannel
FEC	Forward Error Correction
FHC	Frequency Hopping Channel
FIFO	First In First Out
GOP	Group of Pictures
GPS	Generalized Processor Sharing
GSM	Global System for Mobile communications
HIPERLAN/2	HIgh PErformance Radio LAN Type 2
IBBP	Interrupted Batch Bernoulli Process
IDC	Index of Dispersion for Counts
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAN	Local Area Network
MAC	Medium Access Control
MFHC	Multiple Frequency Hopping Channel
MMBBP	Markov Modulated Batch Bernoulli Process
MMBP	Markov Modulated Bernoulli Process
MANET	Mobile Ad hoc Network
MT	Mobile Terminal
NMT	Nordic Mobile Telephone
OFDM	Orthogonal Frequency Division Multiplexing
PILC	Performance Implications of Link layer Characteristics
PAN	Personal Area Network
PDU	Protocol Data Unit
QoS	Quality of Service
RCH	Random access CHannel

SBBP	Switched Batch Bernoulli Process
SFQ	Start-time Fair Queuing
SRPB	Selective Repeat ARQ with Partial Bitmaps
TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UDP	User Datagram Protocol
UL	UpLink
UMTS	Universal Mobile Telecommunication System
U-PDU	User Protocol Data Unit
WFQ	Weighted Fair Queuing
WLAN	Wireless Local Area Network
WWW	World Wide Web

List of Figures

List of Tables

Chapter 1

Introduction

We are experiencing a major paradigm-shift in the telecommunications industry. People are increasingly making use of data applications in addition to the traditional telephony service. A number of popular applications have emerged like email, the World Wide Web and so on, mostly relying on the TPC/IP family of protocols today. Traffic volume of these new data applications is increasing at a much faster rate than traffic volume of telephony. In many places data traffic has already exceeded telephony traffic. It is expected that in the near future data traffic will dominate communication networks in general.

From a technology point of view, this has the consequence that communication devices, architectures and protocols optimized for traditional services like telephony are increasingly being replaced by those optimized for data traffic. In this dissertation we contribute to this shift in technology.

Traffic generated by data applications has basically different characteristics from traffic generated by telephony. Voice calls in traditional systems produce static traffic load for the duration of a call. In contrast, data traffic is dynamic, depending on the nature of the application. Telephony system dimensioning techniques developed by Erlang (see e.g., [49]) do not apply in a data networking context [74]. It has been observed that data traffic exhibits burstiness over many time scales [53].

In Chapter 2, we investigate the dynamic nature of data traffic and propose a means of characterizing its variability. For this, we consider a measure called *peakedness* that was introduced in the context of telephony network dimensioning. However, this measure can be generalized and extended so that we can use it for data traffic characterization as well. We show how peakedness can be used in discrete time, discuss a number of practical considerations and apply the peakedness measure to a number of traces taken from measurements. In addition, we also provide a technique that gives a Markovian traffic model to fit the peakedness of the measured traffic. We show examples when such a model can capture the peakedness of the traffic over many time scales. But we also observe that in some cases we can fit a model at a given time scale only, and not capturing long range dependence in the traffic.

In the rest of the dissertation, we consider different design aspects of mobile data networking architectures. The shift towards mobile networking represents another paradigm-shift in the communications industry. Designers of mobile data networks must face with a number of conflicting requirements. On the one hand, the mobile network must support the prevailing network and upper layer protocols that are in use. Today this implies the support of the TCP/IP family of protocols [44, 85]. However, the TCP/IP protocols and most of the applications for these protocols were designed with a fixed network in mind. The designer of the mobile data network therefore has to facilitate a smooth migration of the TCP/IP protocols from a fixed infrastructure to the mobile system. To enable this migration, the designer of the mobile system seeks to hide

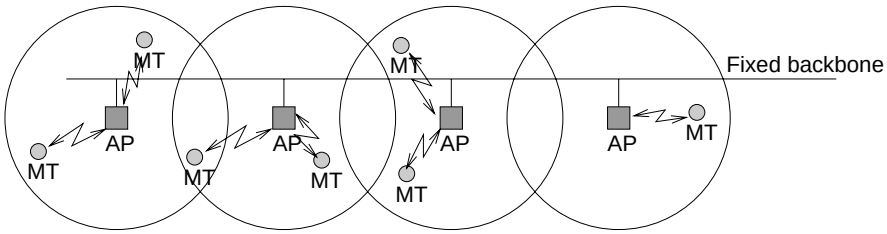


Figure 1.1: Cellular architecture in a wireless LAN

the effects and specifics of the mobile architecture from the upper layers as much as possible.

On the other hand, the mobile network imposes a number of specific requirements and constraints not present in fixed networks. These include the support of user mobility and the fact that wireless transmission links have fundamentally different performance characteristics than wireline transmission links. These considerations call for a set of mechanisms and protocols that are tailor-made for the characteristics of the mobile data networks.

In Chapters 3-5 we address some of these design questions within the context of cellular networks. This is the architecture used by most of today's commercial and proposed standards: first generation mobile systems (including NMT, Nordic Mobile Telephone), second generation mobile systems (including GSM, Global System for Mobile communications), and third generation mobile systems (UMTS, Universal Mobile Telecommunication System) [77]. Wireless Local Area Networks (WLANs) can also make use of the cellular architecture. Figure 1.1 shows an example of a cellular architecture for a WLAN. Each cell is controlled by a base station referred to as an Access Point (AP). These APs are connected by a fixed backbone network. Mobile Terminals (MTs) communicate with the APs over the air interface. Each MT is associated with a single AP that it communicates with. MTs are free to move within the coverage area of the APs.

The architecture immediately poses a number of problems not present in fixed networks. Mobility implies that a set of mechanism must be provided to route data packets to the current location of the MT. As part of this mechanism, the MT also has to change its AP association when it leaves its cell (handoff). We refer to [92] for an extensive treatment and analysis of these questions.

In the dissertation we look at some of the issues that arise in cellular networks concerning the adaptation to the special properties of the wireless links. We take the example of a wireless LAN system, although the proposed solutions can be extended to other systems as well. The presented mechanisms could be applied in public mobile networks as well as private networks, although this dissertation does not investigate the specific aspects of public networks such as charging, service level agreements, etc.

Chapters 3-5 consider specific link layer design issues. In Chapter 3 we investigate the question of link-layer automatic retransmission (ARQ) protocol. Dynamic traffic patterns and quickly changing error characteristics require that this protocol has to be dynamic as well. We propose a new solution in the HIPERLAN/2 wireless LAN architecture. The proposal is novel due to its dynamic nature: it can adapt the amount of ARQ feedback and the actual content of the ARQ messages based on the traffic and error patterns. Simulation results indicate that the dynamic solution yields higher performance than static solutions.

In Chapter 4 we analyze the performance trade-offs in the resource allocation of a wireless base station. Since traffic patterns cannot be predicted, the base station has to make resource decisions in real-time. The wireless channel however makes the problem fundamentally different from that of resource allocation in a fixed network. The amount of service allocated to a user may

be different from the amount of service that the user actually gets because of location-dependent channel errors that are typical for an air interface. We can compensate for the channel errors, but this affects both the total channel utilization and fairness between the users. We introduce a simple compensation mechanism at the base station, and analyze this trade-off, also taking into account the effect of transport layer (TCP) mechanisms and link layer retransmissions.

In Chapter 5, we extend the analysis of Chapter 4 into a practical scheme. Besides using the compensation for the errors at the wireless channel, we also take into account the properties of the wireless channel to avoid errors. We propose a distributed modular architecture where the wireless channel is monitored by the user terminals that make a decision of their own regarding the use of the channel. The scheduler at the base station is such that it encourages the users to make an efficient use of the resources. We propose a master scheduler and a possible user behaviour algorithm and show by simulation and analysis that the proposed scheme can improve both system utilization and fairness. The architecture requires the co-operation of number of adaptive schemes to work in an environment with dynamic traffic patterns and channel errors: the transport layer adapts to the available capacity; the link layer performs automatic retransmissions in a dynamic fashion, and the resource allocation adapts to channel errors and user behaviour, and uses an adaptive scheme to maximize efficiency.

Finally in Chapter 6 we investigate another type of dynamic environment: we consider an ad hoc network where all nodes may be mobile. We propose a new solution that enables the use of frequency hopping spread spectrum in an ad hoc networking context. For this, devices may need to dynamically switch between different frequency hopping channels. We present an analytic and simulation-based performance analysis of different channel configurations.

We conclude the dissertation in Chapter 7 with a summary of the contributions of the thesis and proposals for future research. In Appendix A we provide the derivations for the formulae in Chapter 2.

1.1 Research Objectives

Traditional telecommunication networks were often designed to satisfy a well-determined set of application requirements. In the case of modern mobile packet data communication networks however, application needs are hard to predict, and traffic patterns are bursty and unpredictable. These make it necessary to use dynamic adaptive schemes. The objective of the dissertation is to design and analyze new control mechanisms to support this technological trend.

Specifically, it is the objective of this dissertation to

- investigate methods that characterize the dynamic (bursty) nature of traffic;
- develop solutions that support the efficient transmission of dynamic (bursty) traffic over a wireless channel;
- analyze the trade-off in the resource allocation of a wireless base station with regard to the errors that occur over the air interface;
- design and analyze new resource allocation mechanisms that can maintain control over the trade-off between fairness and utilization, and at the same time support a practical implementation;
- design and analyze new mechanisms that facilitate mobile ad hoc networking using existing radio technology.

Chapter 2

Peakedness Characterisation of Bursty Traffic

2.1 Introduction

An important experience from measurement studies (including Ethernet, ATM LAN/WAN networks [53, 68, 74]) regarding the nature of data traffic is that traffic exhibits bursty properties over many time scales.

One of the key concepts for capturing the bursty character of data traffic is the scale invariance which is often exhibited by self-similarity. The issue resulted in active research on fractal characterization [53, 68]. So far it is not clear how successfully we can utilise self-similarity from a practical traffic engineering point of view but one thing is for sure: burstiness seems to be the most important yet poorly understood characteristic of traffic in high-speed networks. Our work is motivated by this need. In this chapter we focus on peakedness as one of the most promising candidate measures of traffic burstiness.

The simplest burstiness measures take only the first-order properties of the traffic into account. A set of candidates are the moments of the inter-arrival time distribution. In practice the peak to mean ratio and the squared coefficient of variation are the most frequently used first-order measures [72, C1].

Measures expressing second-order properties of the traffic are more complex. The autocorrelation function, the indices of dispersion [83, 36] and the generalized peakedness [19, 20] are the most well known measures from this class.

Moreover, there are a number of burstiness measures based on different concepts, e.g. we can use burst length measures [72, 84] or parameters of a leaky bucket for burstiness characterization [67]. By the concept of self-similarity the Hurst parameter and other fractal parameters are also candidates for burstiness measures [68, 53].

In this chapter we review the theory of generalized peakedness (Section 2.2) and further develop the basic concept by introducing the generalized peakedness in discrete time. The advantage of this approach is that it allows us to apply the general framework of peakedness for traffic engineering. We provide the computation of peakedness for a number of important discrete time models including the Markov Modulated Batch Bernoulli Process (MMBBP) and the batch renewal process. The relationship between IDC and peakedness is also presented. We discuss the challenges of measuring peakedness in practice. Moreover, we show a technique how Markov modulated traffic models can be fitted to a measured peakedness curve. The practical applicability of peakedness and our modelling technique are demonstrated by examples based on measured MPEG video, aggregated ATM and Ethernet traffic in Section 2.3. The chapter is summarized

in Section 2.4. Derivations for the peakedness formulae are elaborated in Appendix A.

2.2 Peakedness Measures

Peakedness of a traffic stream has been found a useful characterization tool in blocking approximations and in trunking theory [40]. It has been defined as the variance to mean ratio of the number of busy servers in an infinite hypothetical group of servers to which the traffic is offered, where the service times of the servers are independent and exponentially distributed with a common parameter.

2.2.1 Generalized Peakedness

Eckberg [19] extended this definition by allowing arbitrary service time distribution and defined *generalized peakedness* as a functional which maps holding time distributions into peakedness values. For a given complementary holding time distribution $F^c(x) = \mathrm{P}\{\text{holding time} > x\}$, Eckberg defines the peakedness functional $z\{F^c\}$ as the variance to mean ratio of the number of busy servers in a hypothetical infinite group of servers with independent holding times distributed according to F^c. The general definition provides a way to characterize the variability of an arrival stream with respect to a given service system.

Let us have a stationary arrival process S in continuous time with counting function $N(t) =$ the number of arrivals in $(0, t]$ for $t \geq 0$. The mean arrival intensity is denoted by $m = \mathrm{E}\{N(t)\}/t$, which is independent of t due to the stationarity of S.

Arrivals are allowed to come in batches of random size B. We define the batchiness parameter as $b = \mathrm{E}\{B^2\}/\mathrm{E}\{B\}$ which can be shown to be the mean size of a batch that an arbitrary arrival finds itself in. The differential process [17] $\Delta N(t)$ is defined for a fixed Δt as the number of arrivals in $(t, t + \Delta t]$, that is, $N(t + \Delta t) - N(t)$. We define the covariance density of the arrival process $k(s)$ for $s > 0$ as the covariance of the differential process as Δt goes to zero:

$$k(s) = \lim_{\Delta t \to 0} \frac{\mathrm{Cov}\{\Delta N(t), \Delta N(t+s)\}}{(\Delta t)^2} \tag{2.1}$$

which is independent of t due to the stationarity of S. For $s < 0$ we let $k(s) = k(-s)$.

We offer the arrival process S to an infinite server group where the service times are independent and have a complementary holding time distribution of $F^c(x)$ ($x \geq 0$; for $x < 0$, we define $F^c(x) = 0$), mean holding time of

$$1/\mu = \int_{-\infty}^{\infty} F^c(x)dx \tag{2.2}$$

where μ is the service rate, and finally the autocorrelation of F^c is

$$\rho_{F^c}(x) = \int_{-\infty}^{\infty} F^c(s)F^c(s+x)ds. \tag{2.3}$$

Denoting the number of busy servers at time t by $L(t)$, the generalized peakedness functional is defined as

$$z\{F^c\} = \frac{\mathrm{Var}\{L(t)\}}{\mathrm{E}\{L(t)\}}. \tag{2.4}$$

If the arrival stream is defined for the whole time axis $(-\infty, \infty)$, it is independent of t due to the stationarity of S. In practice, we never have an arrival process for an infinitely long time; in this case, we have to define the peakedness for a time period t which is large enough for

the initial transient period in the service system to be negligible. (More precisely, $z\{F^c\} = \lim_{t\to\infty} \mathrm{Var}\{L(t)\}/\mathrm{E}\{L(t)\}$.)

With the notation introduced above, the peakedness of the arrival stream can be expressed in terms of the covariance density function as [19]

$$z\{F^c\} = 1 + \frac{\mu}{m}\int_{-\infty}^{\infty} (k(s) - m\delta(s))\rho_{F^c}(s)ds \tag{2.5}$$

where $\delta(s)$ is the Dirac delta function.

The important case of exponential service time simplifies to

$$z_{\exp}(\mu) = \frac{b+1}{2} + \frac{1}{m}k^*(\mu) \tag{2.6}$$

where $k^*(\mu) = \int_{0+}^{\infty} k(s)e^{-\mu s}ds$, the Laplace transform of the covariance density function. Here we have the peakedness of a given arrival stream as a function of the service rate μ.

It is shown [19] (and is suggested by eq. (2.6)) that the peakedness function $z_{\exp}(\mu)$ together with m determines $k(s)$ and therefore the pair $(z_{\exp}(\mu), m)$ is a complete second order characterization of the arrival process.

The peakedness function $z_{\exp}(\mu)$ can be used to compute the peakedness functional for a large class of holding time distributions as shown in [19]. The method is elaborated in [66] to give the peakedness functional for Coxian holding time distributions. The importance of Coxian holding times lies in the fact that any holding time distribution can be approximated with arbitrary accuracy by Coxian distributions. Eckberg also investigated the application of generalized peakedness in delay systems [20]. Eckberg's definition of generalized peakedness for point processes has been extended in [62, 63] to allow fluid flow models given by a rate function.

2.2.2 Peakedness in Discrete Time

In order to use the peakedness measures in a data networking framework, we now extend the peakedness concept for discrete time arrival streams.

We use the following notation: for all $i = \ldots, -1, 0, 1, \ldots$, the number of arrivals at epoch i is $w[i]$. We assume the stationarity of $w[i]$. The first and second moments of $w[t]$ (independent of t) are denoted by m_1 and m_2. The covariance density of continuous time is replaced here by the autocovariance function $k[s] = \mathrm{Cov}\{w[i], w[i+s]\} = k[-s]$. (It is seen that $k[0] = m_2 - m_1^2$.)

The service time random variable T is also discrete and has the distribution $t[1], t[2], \ldots$ on positive integers. (It cannot take on zero value.) $\mu = 1/\mathrm{E}\{T\}$ is again the service rate, and it is easily shown that

$$1/\mu = \mathrm{E}\{T\} = \sum_{s=-\infty}^{\infty} F^c[s] \tag{2.7}$$

where $F^c[x]$ is the complementary holding time distribution function: $F^c[x] = \sum_{u=x+1}^{\infty} t[u] = \mathrm{P}\{T > x\}$ if $x \geq 0$ and $F^c[x] = 0$ if $x < 0$. The autocorrelation function is now

$$\rho_{F^c}[x] = \sum_{s=-\infty}^{\infty} F^c[s]F^c[s+x]. \tag{2.8}$$

It is seen that $\rho_{F^c}[0] = \sum_{s=-\infty}^{\infty}(F^c)^2[s]$.

The traffic is offered to an infinite group of servers with independent identically distributed service times determined by $F^c[x]$. Each arrival takes a separate server. The peakedness of the arrival stream is defined as the variance to mean ratio of the number of busy servers in the infinite

server group:

$$z\{F^c\} = \frac{\mathrm{Var}\{L[t]\}}{\mathrm{E}\{L[t]\}} \tag{2.9}$$

where $L[t]$ is the number of busy servers at time epoch t.

An important modification of the definition is to let the service time depend on the arrival epoch only (have a common service time for all $w[t]$ arrivals at epoch t). We call (in accordance with [63]) the peakedness value defined in this way the *modified* peakedness $\tilde{z}\{F^c\}$. As shown in Section A.3,

$$\tilde{z}\{F^c\} - z\{F^c\} = \left(\frac{m_2}{m_1} - 1\right)(1 - \rho_{F^c}[0]\mu). \tag{2.10}$$

that is, their difference is constant (cf. (35) in [63]). The first factor in the difference is zero if and only if the arrival stream has no simultaneous arrivals, the second factor is zero if and only if the holding time distribution is deterministic.

The importance of this modified definition lies in the fact that it gives a way to handle a whole batch of arrivals together, which can save a lot of computational effort in the case of measuring the peakedness for a general holding time distribution. However, in the case of geometric service times, the original definition of peakedness is easier to measure as shown in Section 2.3.1. We will use the original definition of peakedness (eq. (2.9)) below.

We can express peakedness in terms of the autocovariance function $k[s]$ similarly to eq. (2.5) as

$$z\{F^c\} = 1 + \frac{\mu}{m_1} \sum_{s=-\infty}^{\infty} \rho_{F^c}[s](k[s] - m_1\delta[s]). \tag{2.11}$$

The most important case in discrete time is the case of geometrically distributed holding times: $t[i] = \mu(1-\mu)^{i-1}$, $0 < \mu < 1$ (with $\mathrm{E}\{T\} = 1/\mu$ which justifies the notation).

In order to simplify the formulas, let us introduce the notation

$$K[s] = \begin{cases} \frac{2}{m_1}k[s] & \text{if } s > 0 \\ \frac{1}{m_1}k[0] & \text{if } s = 0 \end{cases}$$

and let its z-transform be $K^*(\omega) = \sum_{s=0}^{\infty} K[s]\omega^s$.

The peakedness function of the arrival stream with respect to geometric holding time distribution, as derived in Section A.4 is given by

$$z_{\mathrm{geo}}(\mu) = 1 + \frac{K^*(1-\mu) - 1}{2 - \mu} \tag{2.12}$$

2.2.3 Peakedness and Index of Dispersion for Counts

The widely used measure to characterize the variability of an arrival stream on different time scales is the index of dispersion for counts (IDC). It is defined as

$$I[t] = \frac{V[t]}{E[t]} = \frac{V[t]}{m_1 t} \tag{2.13}$$

where $E[t]$ and $V[t]$ are the mean and variance of the number of arrivals in t consecutive epochs ($t = 1, 2, \ldots$).

The connection of IDC and peakedness for geometric holding times is (see Section A.5)

$$z_{\mathrm{geo}}(\mu) = 1 - \frac{\mu^2 \frac{dI^*(1-\mu)}{d\mu} + 1}{2 - \mu} \tag{2.14}$$

where $I^*(\omega)$ is the z-transform of $I[t]$.

We can use eq. (2.14) to get asymptotic results which connect them: (see Section A.5):

$$z_{\text{geo}}(0) = \frac{\lim_{s\to\infty} I[s] + 1}{2} \tag{2.15}$$

$$z_{\text{geo}}(1) = I[1] = \frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}} \tag{2.16}$$

where the first equation is derived using the final value theorem and the L'Hospital rule (assuming that $\lim_{s\to\infty} I[s]$ exists and is non-zero), whereas the second equation is derived by the initial value theorem for $\frac{d}{d\omega}I^*(\omega)$.

2.2.4 Peakedness of Traffic Models

Next, we present the peakedness results for important traffic models. We consider discrete time models for the number of arrivals in consecutive epochs.

Batch Bernoulli Process

A very simple type of arrival stream model is the model with the number of arrivals in a time epoch be independent identically and generally distributed with mean m_1 and second moment m_2 (Batch Bernoulli Process, BBP).

In this case, $k[i] = 0$ for all $i > 0$. Thus,

$$K^*(1-\mu) = K[0] = \frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}} \tag{2.17}$$

and

$$z_{\text{geo}}(\mu) = 1 + \frac{\frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}} - 1}{2-\mu} \tag{2.18}$$

For the special case of Poisson batch arrivals, the distribution of arrivals in an epoch is Poissonian, thus

$$\frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}} = 1 \tag{2.19}$$

which gives $z_{\text{geo}}(\mu) = 1$.

The Poisson process can be considered as a reference process with respect to peakedness characterization. Batch arrival processes that are more bursty than the Poisson process have higher peakedness values, smoother processes have lower peakedness. (In the case of deterministic traffic, $z_{\text{geo}}(\mu) = 1 - \frac{1}{2-\mu}$.)

Markov Modulated Batch Bernoulli Process

A very general Markovian process is the Markov Modulated Batch Bernoulli Process (MMBBP). In this model, we have a discrete time Markov process as a modulating process. In each state of the modulating Markov-process, batch arrivals are generated according to a general distribution corresponding to the state.

Let $\mathbf{P}$ and $\mathbf{D}$ denote the transition probability matrix and the steady-state distribution vector of the modulating Markov process, respectively ($\mathbf{DP}=\mathbf{D}$). Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be diagonal matrices corresponding to the first and second moments of the number of arrivals in the corresponding states. Let $\mathbf{e}$ be a vector of all ones and let $\mathbf{I}$ be the identity matrix.

We can express the mean number of arrivals as $m_1 = \mathbf{D}\mathbf{M_1}\mathbf{e}$ and the second moment as $m_2 = \mathbf{D}\mathbf{M_2}\mathbf{e}$. The autocovariance function of the arrival process is given by $k(i) = \mathbf{D}\mathbf{M_1}\mathbf{P}^i\mathbf{M_1}\mathbf{e} - m_1^2$.

Using eq. (2.12) we have derived the peakedness function (see Section A.6) as

$$z_{\text{geo}}(\mu) = 1 + \frac{1}{2-\mu}\left(\frac{2(1-\mu)\mathbf{D}\mathbf{M_1}\mathbf{P}(\mathbf{I}-(1-\mu)\mathbf{P})^{-1}\mathbf{M_1}\mathbf{e} + m_2}{m_1} - 1\right) - \frac{m_1}{\mu} \tag{2.20}$$

A very important case of MMBBP is the Markov Modulated Bernoulli Process (MMBP); its peakedness curve is the special case of eq. (2.20).

Markov Modulated Bernoulli Process

As a special case of MMBBP, when the arrival process is Bernoulli in each state, we have a Markov Modulated Bernoulli Process (MMBP). If the parameter of the Bernoulli process is p_i in state i, we have $\mathbf{M_1} = diag(p_1, p_2, \ldots...)$ and $\mathbf{M_2} = \mathbf{M_1}$.

The peakedness curve for geometrical holding times in this case is

$$z_{\text{geo}}(\mu) = 1 + 2\frac{(1-\mu)\mathbf{D}\mathbf{M_1}\mathbf{P}(\mathbf{I}-(1-\mu)\mathbf{P})^{-1}\mathbf{M_1}\mathbf{e}}{(2-\mu)m_1} - \frac{m_1}{\mu} \tag{2.21}$$

Switched Batch Bernoulli Process

Another important special case of MMBBP is the 2-state MMBBP (SBBP, Switched Batch Bernoulli Process). Let us use the following notation: the transition matrix is

$$\mathbf{P} = \begin{bmatrix} 1-\alpha_1 & \alpha_1 \\ \alpha_2 & 1-\alpha_2 \end{bmatrix} \tag{2.22}$$

and the steady state distribution is thus

$$\mathbf{D} = \frac{1}{\alpha_1+\alpha_2}(\alpha_2\ \alpha_1). \tag{2.23}$$

Denote $\gamma = 1-\alpha_1-\alpha_2$. In state 1, the first and second moments of the number of arrivals are $m_{1,(1)}$ and $m_{1,(2)}$, respectively; in state 2, the moments are $m_{2,(1)}$ and $m_{2,(2)}$.

The first and second moments of the number of arrivals are given by

$$m_1 = \frac{1}{\alpha_1+\alpha_2}(\alpha_2 m_{1,(1)} + \alpha_1 m_{2,(1)}), \tag{2.24}$$

$$m_2 = \frac{1}{\alpha_1+\alpha_2}(\alpha_2 m_{1,(2)} + \alpha_1 m_{2,(2)}). \tag{2.25}$$

Let us also introduce the notation

$$m_* = \frac{1}{\alpha_1+\alpha_2}(\alpha_2 m_{1,(1)}^2 + \alpha_1 m_{2,(1)}^2). \tag{2.26}$$

Note that if the distribution of the batch size in a given state is deterministic, or if it is geometric or Bernoulli, we have $m_{i,(1)}^2 = m_{i,(2)}$ $(i = 1, 2)$ and thus $m_* = m_2$. If the batch distribution is Poisson, we have $m_* + m_1 = m_2$.

Using eq. (2.20) and the possibility to explicitly compute the inverse of $\mathbf{I} - (1-\mu)\mathbf{P}$ in the 2-state case, we get

$$z_{\text{geo}}(\mu) = 1 + \frac{1}{2-\mu}\left(\frac{2}{m_1}\frac{(1-\mu)}{\mu}\left[m_* - \frac{(m_* - m_1^2)(1-\gamma)}{1-\gamma(1-\mu)}\right] + \frac{m_2}{m_1} - 1\right) - \frac{m_1}{\mu} \tag{2.27}$$

and by eq. (2.12) we get the peakedness curve.

It is interesting and important to note that the peakedness curve depends on the SBBP parameters only through m_1, m_2, m_*, γ. Therefore, we can get identical peakedness values for different SBBPs if these four parameters coincide.

Batch Renewal Process

The batch renewal process is important to consider because of its ability to model the correlation structure of traffic [51]. The discrete time batch renewal process is made up of batches of arrivals, where the intervals between batches are independent and identically distributed random numbers, and the batch sizes are also independent and identically distributed, furthermore, the batch sizes are independent from the intervals between batches.

We use the following notation for the discrete time batch renewal process: a and b are the mean length of intervals between batches and the mean batch size, respectively. The first and second moments of the number of arrivals in an epoch is given by $m_1 = b/a$, and $m_2 = m_1 b(C_b^2 + 1)$ where C_b^2 is the squared coefficient of variation (variance to mean square ratio) of the batch size. The probability generating function of the distribution of time between batches is denoted by $A^*(\omega)$. ($A^*(\omega) = \sum_{s=1}^{\infty} a[s]\omega^s$ where $a[s]$ is the probability that the time between two consecutive batches is s.)

The peakedness for geometric holding times is given by

$$z_{\text{geo}}(\mu) = 1 + \frac{1}{2-\mu}\left(\frac{1 + A^*(1-\mu)}{1 - A^*(1-\mu)} - b + \frac{m_2}{m_1} - 1\right) - \frac{m_1}{\mu}. \tag{2.28}$$

If the distribution of time between batches follows a shifted generalized geometric distribution [51], that is, $a[t] = 1 - \sigma$ if $t = 1$ and $a[t] = \sigma\tau(1-\tau)^{t-2}$ if $t = 2, 3, \ldots$, then its probability generating function is:

$$A^*(\omega) = \omega\left(1 - \sigma + \frac{\sigma\tau\omega}{1 - (1-\tau)\omega}\right) \tag{2.29}$$

which makes the peakedness values easily computable.

2.2.5 Fitting Traffic Models to Peakedness Curves

The peakedness shows the variability of the arrival stream with respect to different service holding times. It is of interest to investigate whether we can fit traffic models to peakedness curves based on measurements.

We outline here a fitting procedure based on the mean rate m_1 of the arrival traffic, the peakedness value at $\mu = 1$ and at three other points, μ_1, μ_2, μ_3. The model we fit to the peakedness curve is an Interrupted Batch Bernoulli Process (IBBP): in one state of the modulating Markov process, the arrival number has a general distribution, in the other state, there are no arrivals. The fitting is done by reversing the peakedness function of the Markov Modulated Batch Bernoulli Process of Section 2.2.4.

First, by $z(1) = m_2/m_1 - m_1$, we get m_2. Introducing $\omega = 1 - \mu$, $\omega_i = 1 - \mu_i$ and using the notations of Section 2.2.4, we can compute (using the values $K^*(\omega_i) = (z_{\text{geo}}(\mu_i) - 1)(\omega_i + 1) + 1$)

$$Y_i = Y(\omega_i) = m_1 \frac{1-\omega_i}{2\omega_i}\left(K^*(\omega_i) + m_1 \frac{1+\omega_i}{1-\omega_i} - \frac{m_2}{m_1}\right) \tag{2.30}$$

Using eq. (2.27),

$$Y(\omega) = m_* - \frac{(m_* - m_1^2)(1-\gamma)}{1 - \gamma\omega}. \tag{2.31}$$

Let us denote

$$\tilde{Y} = \frac{Y_1 - Y_2}{Y_2 - Y_3} \tag{2.32}$$

which evaluates to

$$\tilde{Y} = \left(\frac{\omega_2 - \omega_1}{\omega_3 - \omega_2}\right)\left(\frac{1 - \gamma\omega_3}{1 - \gamma\omega_1}\right) \tag{2.33}$$

and we get

$$\gamma = \frac{\tilde{Y}\frac{\omega_3 - \omega_2}{\omega_2 - \omega_1} - 1}{\tilde{Y}\frac{\omega_3 - \omega_2}{\omega_2 - \omega_1}\omega_1 - \omega_3} \tag{2.34}$$

Once we have γ, we can obtain an estimation for m_* as

$$m_* = \frac{1}{3}\sum_{i=1}^{3} \frac{Y_i - \frac{m_1^2(1-\gamma)}{1-\gamma\omega_i}}{1 - \frac{1-\gamma}{1-\gamma\omega_i}} \tag{2.35}$$

where we have on the right hand side an average for the known values ω_i, Y_i.

Then it is possible to fit an IBBP as follows:

$$m_{1,(1)} = \frac{m_*}{m_1}, \alpha_2 = \frac{m_1(1-\gamma)}{m_{1,(1)}}, \alpha_1 = 1 - \gamma - \alpha_2, m_{1,(2)} = m_2\frac{\alpha_1 + \alpha_2}{\alpha_2}. \tag{2.36}$$

Given the first and second moments of the number of arrivals in state 1, we can use for example a generalized geometric distribution for modelling the batch size distribution. In this case, there are no arrivals with probability $1 - \varphi$, and there is a batch of arrivals with geometrically distributed size of parameter ψ. The moments are given by

$$m_{1,(1)} = \varphi/\psi, \tag{2.37}$$

$$m_{1,(2)} = \varphi/\psi^2 \tag{2.38}$$

by which we can get φ, ψ for the model.

If it is possible to exactly fit an IBBP to the μ_i, $z_{\text{geo}}(\mu_i)$ pairs, the values that are summed in the equation for m_* are identical. If there is no IBBP that exactly fits the given peakedness values, m_* gives an estimation and the peakedness curve of the fitted IBBP model approximates the μ_i, $z_{\text{geo}}(\mu_i)$ pairs.

2.3 Generalized Peakedness of Real Traffic

2.3.1 Measuring Peakedness

To measure the generalized peakedness of a traffic with a given holding time distribution, one can simulate the infinite server group. In discrete time, one can keep track of the first and second moments of the number of busy servers and compute the variance to mean ratio from them. The following points should be made about the estimation.

- We should take care of the initial phase of the simulation. If we have no prior knowledge about the traffic, we do not know what the mean number of busy servers will be. In this case, we can start from an empty system. The initial transient in the number of busy servers should be excluded from measurements.

- According to the definition, we should assign a server to each arrival, that is, assign a random holding time variable to every arrival in an epoch, which could involve a huge

amount of computational effort. However, using the modified definition of peakedness and eq. (2.10), we can reduce the computational effort by assigning only one random service time variable to all arrivals in an epoch.

- When the service time is geometric, we can minimize the computational effort by making use of the memoryless property. If at epoch t we have $L[t]$ busy servers, then at the next epoch we have $L[t+1] = L[t] + w[t+1] - D[t]$ where $D[t]$ is the number of departures from the service system at epoch t.

  The distribution of $D[t]$ is known to be binomial with parameters $L[t]$ and μ because each of the $L[t]$ servers finish service with probability μ. Therefore, in the measurement, it is enough to keep track of $L[t]$ together with the first and second moments of the previous $L[i], i \leq t$ values.

  This gives us the following procedure for computing the peakedness value for geometric holding time distribution with parameter μ:

  1. Reset $L_1 = 0$, $L_2 = 0$, L_{old} =initial value (see comments below);
  2. Set $L_{new} = L_{old} + w_{new} - d$ where d is a random number of distribution $binom(L_{old}, \mu)$ and w_{new} is the number of new arrivals in the next epoch;
  3. Set $L_1 = L_1 + L_{new}$, $L_2 = L_2 + L_{new}^2$;
  4. Set $L_{old} = L_{new}$ and loop back to 2 unless the measurement is over;
  5. Compute $l_1 = L_1/T, l_2 = L_2/T, z = l_2/l_1 - l_1$ where T is the length of the total measurement time.

  The setting of the initial value of L_{old} depends on the amount of a priori information that we have about the traffic. If we know the mean rate, we can set the initial L_{old} to its mean value determined by Little formula as m_1/μ. If we do not know the mean rate, we have to start from an empty system (initial $L_{old} = 0$) and simulate the service system without actually measuring (executing step 3) until the initial transient is over.

- An important advantage of using peakedness characterization is that we can measure peakedness by going through the traffic trace in only one sequence. This gives us the possibility of measuring peakedness for real-time traffic on the fly.

  Computing peakedness for one value of μ involves N cycles of the above procedure (where N is the total length of the measured traffic); if we want to measure peakedness at several μ values, we can easily implement the parallel execution of the procedure. In each cycle, we only have to compute a small number of additions and multiplications, and generate one binomially distributed random variable. Therefore, the complexity of the measurement is $O(N)$. The most time-consuming step in the measurement is the generation of the binomially distributed random number. We can reduce the computational cost of the measurement tremendously by approximating it with a normally distributed random number, for which pre-computed look-up tables can be used.

- It is interesting to note that the measurement of peakedness involves randomness due to the simulation of servers, which means that for one sequence of traffic we could get different peakedness values. Experiments show that the peakedness measurements do not change significantly if we compute them more than once.

- The advantage of our approach compared to Eckberg's method for estimating peakedness for exponential holding times (cf. [20, 63]) is that our method does not neglect a lot of arrivals in the computation due to the selection of an arbitrary arrival.

2.3.2 Peakedness of Video Traffic

Video traffic is a very important example of variable rate traffic. We investigated the application of peakedness measure for the characterization of variability of MPEG video traces [79].

In the MPEG coding scheme, the sequence of frames are divided into Groups of Picture (GOP), where each GOP is made up of so-called I, P and B frames. I frames are the largest because no prediction is used for coding them; P and B frames are smaller because one and two-directional prediction decreases the amount of information to be coded. The MPEG sequences that we considered had a GOP (Group of Pictures) length of 12 frames, a GOP pattern of IBBPBBPBBPBB, and frames capture frequency of 25 frames per second.

Figure 2.1 shows the peakedness curve of an MPEG video trace of a movie (MrBean) as a function of the service rate μ. The mean service time of a server is therefore $1/\mu$ time epochs, where one time epoch is now 40ms. The solid curve is the peakedness function for the frame sequence (one frame corresponds to one epoch), whereas the dashed curve is the peakedness function for the GOP sequence (one GOP corresponds to 12 epoch so that is has the same time-length as the frame sequence) The scaling in the vertical axis is such that one arrival corresponds to one bit.

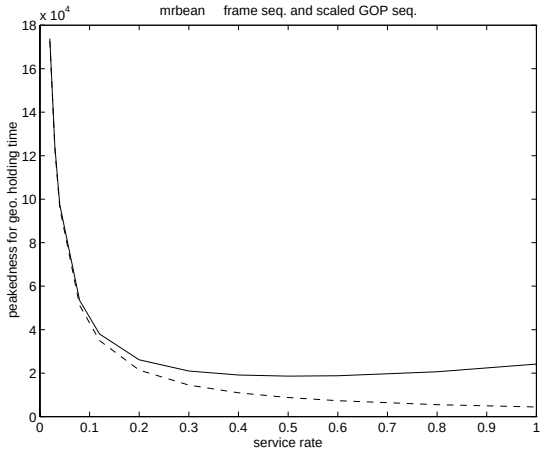


Figure 2.1: Peakedness of the frame (solid) and GOP (dashed) sequence of MPEG video trace (MrBean).

By decreasing the service rate, the service times become longer, and the number of busy servers in the infinite server group depends on the traffic properties on longer time scales. In this way, the peakedness curves show the variability of the traffic on different time scales, i.e. on the time scale of $1/\mu$.

Figure 2.1 shows that on short time scales, the variability of the frame sequence is much greater compared to the GOP sequence. But as we go to longer and longer time scales, the variability of the two sequences converge. What we can learn from this is that on longer time scales (for example, when dimensioning larger buffers), the statistical characteristics of GOP structure is less significant, and it is enough to consider the GOP sequence.

Figure 2.2 shows the peakedness curves for geometric service time distributions for five MPEG video GOP size traces. It gives us a relative comparison of the variability of different kinds of video sequences. (In this figure, one time epoch is set to one GOP which introduces a scaling compared to Figure 2.1.) The highest values of peakedness are exhibited by the MTV sequence,

which is known to have lots of scene changes. Movie sequences show lower peakedness compared to the MTV sequence. The peakedness of a video conference sequence is found to be the smallest by orders of magnitude.

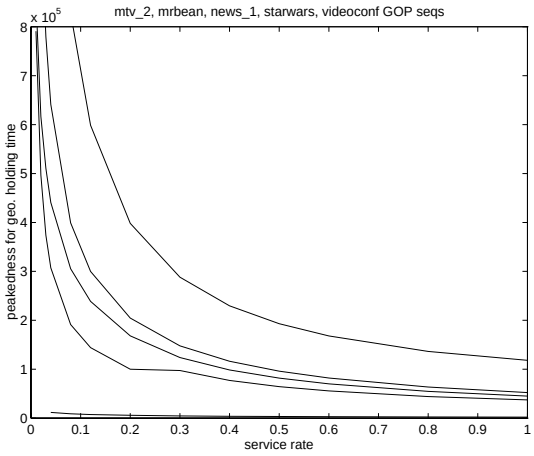


Figure 2.2: Peakedness of MPEG GOP video sequences. From the uppermost downwards, the sequences are from: TV (MTV), movie (MrBean), TV (News), movie (StarWars), video conference.

Figure 2.3 shows an IBBP fitted to an MPEG movie trace (MrBean, [79]). The solid line is the peakedness curve of the GOP sequence, the dashed line shows the peakedness curve of the fitted model. The circles show the peakedness values where the fitting was made. The points were chosen to represent the variability of the traffic on a long time scale (corresponding to the time scale of 1/0.01=100 epoch, here one epoch corresponds to 0.48 sec). As we can see, the model is able to capture the variability of the arrival stream on the investigated time scales.

2.3.3 Peakedness of Aggregated ATM Traffic

We analysed the peakedness curve of an aggregated ATM traffic trace taken from the Finnish University and Research ATM WAN network (FUNET) [68]. The trace was approximately one hour long and consisted of the number of cell arrivals in each second. Figure 2.4 shows the peakedness curve of the measurement and two IBBPs fitted to it. The IBBP that was fitted at short time scale fits the measured peakedness curve well for shorter time scales, but it gives lower peakedness values for time scales longer than $1/0.05 = 20$sec. The other IBBP was fitted at a longer time scale; this model gives lower peakedness values for time scales shorter than 20sec.

2.3.4 Peakedness of Ethernet Traffic

Figure 2.5 and Figure 2.6 show the peakedness curve of an Ethernet traffic taken from the Bellcore measurements [53]. The measurement covers 1 million arrivals (approx. one hour). Figure 2.5 depicts peakedness on a lin-lin plot, Figure 2.6 is a log-log plot. We can investigate 5 different time scales in Figure 2.6. The interesting finding is that the peakedness increases linearly on the log-log plot as we decrease the rate (go to long time scales). Due to eq. (2.15) and knowing that $\lim_{s\to\infty} I[s] = \infty$ if there is long range dependence (LRD) in the traffic, the peakedness diverges as the rate goes to zero. This observation of monotonicity in Figure 2.6 supports the presence

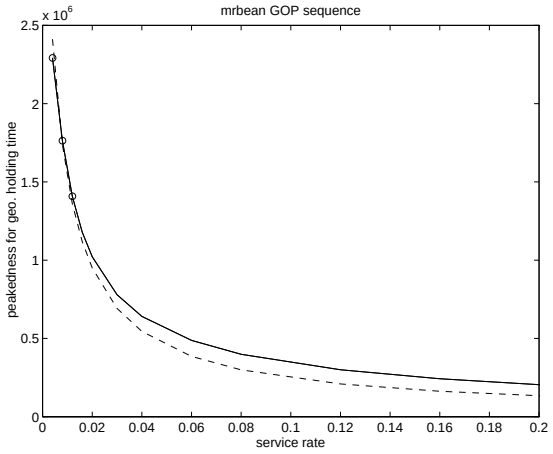


Figure 2.3: Peakedness curves of MPEG GOP movie trace (MrBean, solid) and its IBBP model (dashed).

of LRD assuming that the traffic stationarity assumption holds. It is important to note that *the peakedness curve can be used as an indicator of LRD.*

At different time scales we fitted simple Markovian models (IBBPs) to capture the peakedness curves in Figure 2.6. We can see that the burstiness scaling property of these models are not appropriate i.e. these models can cover a shorter range of time scales in burstiness than it would be necessary to follow the burstiness of the real traffic over all the investigated time scales.

Our investigations of the aggregated ATM and Ethernet traffic indicate that simple Markovian models are not able to capture the burstiness characteristic of traffic over many time scales. For this case fractal traffic models seem to be more appropriate [68, 53]. However, for several practical cases we do not need to focus on *all* time scales but only on our working time scales (e.g. time scales of queuing) which can be efficiently modelled by Markovian models, too.

2.4 Summary

We have shown that peakedness can be used to characterize the bursty nature of traffic. Peakedness curves show the variability of traffic on different time scales and can be efficiently computed for real time traffic. We have extended the peakedness theory to discrete time and applied the peakedness characterization to variable rate video traffic, Ethernet traffic and aggregated ATM traffic as well as to the most important traffic models. We have shown that generalized peakedness can also be used for detecting long range dependence. We have also presented a new model fitting technique based on the concept of peakedness.

The basic idea of peakedness characterization is that we characterize traffic by its interactions with the service system. Although the traffic is usually offered to more complicated queuing systems, it is difficult to use complicated systems for characterization because it is very hard to handle them analytically. The infinite server group may be regarded as a compromise between generality and analytical tractability. Its generality is shown by the observation that peakedness gives a complete second order characterization, i.e. it contains all information about the correlation structure of the traffic.

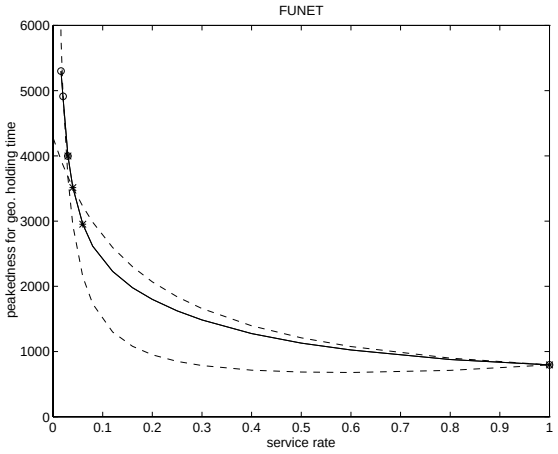


Figure 2.4: Peakedness of aggregated ATM traffic (solid) and IBBP models (dotted) fitted to it. The two IBBPs are fitted at short (stars) and long (circles) time scales.

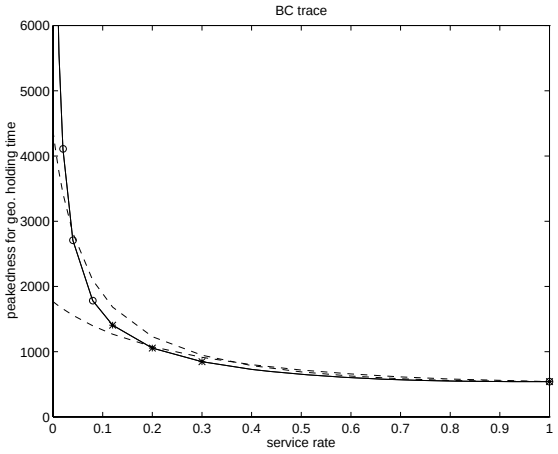


Figure 2.5: Peakedness of Ethernet trace (solid) and IBBP models (dotted) fitted to it. The two IBBPs are fitted at short (stars) and long (circles) time scales.

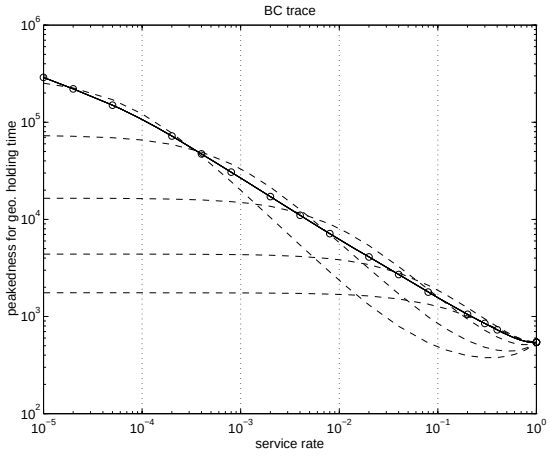


Figure 2.6: Peakedness of Ethernet trace (solid) in log-log plot. On five time scales (separated by vertical lines) IBBP models are fitted (dashed).

Chapter 3

Design of a New Dynamic Retransmission Protocol

3.1 Introduction

Mobile networks have to cope with the special properties of the wireless links between a terminal and the base station. Wireless links have typically a much higher error rate than fixed links. In addition, the error patterns and other performance characteristics such as the capacity of the link can change dynamically over time. To cope with these problems, the wireless network has to use a set of dynamic error control mechanisms to compensate the performance effects of the wireless links.

This chapter discusses this issue, namely the dynamic error control in the communication between the AP (Access Point) and MT (Mobile Terminal) (as shown earlier in Figure 1.1). We will address this question within the context of the HIPERLAN/2 wireless LAN system, but the presented solutions can be applied to other architectures as well.

The chapter is organized as follows. We first give a brief description of the HIPERLAN/2 architecture. We review related work on retransmission (ARQ) protocols and show how they could be applied in the error control mechanism of HIPERLAN/2 in Section 3.3. We then discuss a new dynamic ARQ protocol for dynamic error control in Section 3.4. The chapter is summarized in Section 3.5.

3.2 The HIPERLAN/2 Architecture

HIPERLAN/2 stands for HIgh PErformance Radio LAN type 2 [37, 45, 47, 46], a Wireless Local Area Network (WLAN) system standardized by ETSI/BRAN (European Telecommunication Standards Institute Broadband Radio Access Networks project, [28]). Wireless LANs provide high-speed wireless packet-oriented connectivity in a local coverage area [33]. Below we give a brief overview of the HIPERLAN/2 standard, which will help in understanding the ARQ protocols later in this chapter, and also in understanding the subsequent chapters. However, the deep understanding of the HIPERLAN/2 system details is not essential for the purposes of this dissertation.

The HIPERLAN/2 standard uses a cellular architecture (see Figure 1.1) and operates in the 5GHz band. It enables users to access Ethernet, IP, UMTS, ATM or other types of backbone networks with high data rate (up to 54Mbps) and can provide different levels of QoS. The standard covers the physical and data link layer, which we briefly summarize below.

Mode	Modulation	Code rate	Physical layer bit rate
1	BPSK	1/2	6 Mbps
2	BPSK	3/4	9 Mbps
3	QPSK	1/2	12 Mbps
4	QPSK	3/4	18 Mbps
5	16QAM	9/16	27 Mbps
6	16QAM	3/4	36 Mbps
7	64QAM	3/4	54 Mbps

Table 3.1: Physical layer modes of HIPERLAN/2

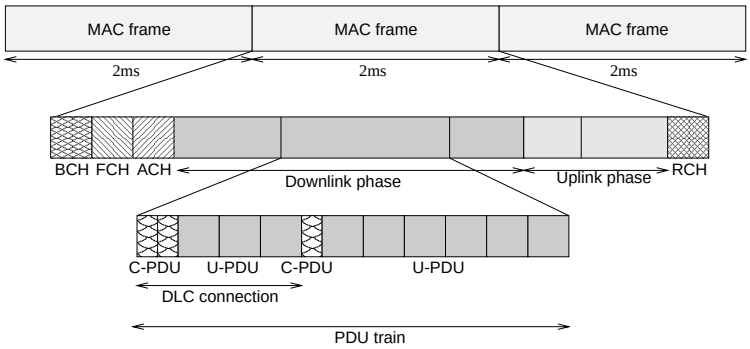


Figure 3.1: Basic MAC frame structure in HIPERLAN/2

In the physical layer of HIPERLAN/2, a transmission scheme called Orthogonal Frequency Division Multiplexing (OFDM) has been selected due to its excellent performance in an indoor environment with multiple paths and reflections present between the transmitter and receiver [48]. A key feature of the physical layer is to provide several modes with different coding and modulation schemes (see table 3.1), which are selected adaptively by a link adaptation algorithm based on measurements of the channel [88]. Choosing the appropriate link adaptation algorithm enables the system to match the physical layer mode to the required radio link quality in order to reach the desired quality of service (e.g., maximum throughput or low delay).

The physical layer implements another adaptation mechanism, namely dynamic frequency selection (DFS) [88]. This adaptation works on a long time scale so that it does not react to quick changes in the traffic or error pattern. The DFS mechanism provides an automatic way for each base station to allocate the frequency used in its cell. Manual frequency planning is therefore avoided, and the system can automatically react to the installation or uninstallation of other base stations, or to the appearance or disappearance of other interferers in the vicinity.

In the data link control (DLC) layer, a new level of adaptation is possible by dynamically scheduling traffic based on the current traffic demand. This is made possible by the air interface that is based on dynamic time division multiple access (TDMA) time division duplex (TDD) MAC (Medium Access Control) frame structure. In order to achieve a high utilization of radio channel capacity, the access points (APs) and mobile terminals (MTs) in the system dynamically share the transmission capacity.

The dynamic sharing of capacity is achieved by a MAC frame structure that is controlled by a scheduling algorithm in the AP and adapts to traffic demands. The MAC frame appears with

a period of 2 ms and has the following basic structure (Figure 3.1). At the beginning of a MAC frame there is a broadcast channel (BCH, sent by the AP) that conveys information concerning the whole cell and indicates the start point and the length of some elements of the MAC frame, e.g., location of random access channels (RCHs). The frame channel (FCH, sent by the AP) contains an exact description of how resource has been allocated within the MAC frame in the downlink (from AP to MT) and uplink (from MT to AP) phase. The access feedback channel (ACH, sent by the AP) conveys information on previous access attempts made in the RCH. The ACH is followed by the downlink and uplink phases, where the border between downlink and uplink can dynamically change from frame to frame. The downlink and uplink phases consist of PDU trains to and from MTs. A PDU train comprises user PDUs (U-PDUs of 54 bytes with 48 bytes of payload) and control PDUs (C-PDUs of 9 bytes) to be transmitted or received by one MT. There is one PDU train per MT (if resource has been granted in the FCH). As can be seen in Figure 3.1, a MT can have traffic over multiple DLC connections in one PDU train. Each DLC connection corresponds to a specific QoS support on the DLC layer. The scheduler in the AP is responsible for sharing transmission capacity between MTs and AP in uplink and downlink based on the needed QoS. At the end of the frame, the RCH provides a contention period that can be used by the MTs to request transmission resources for the uplink phase in upcoming MAC frames, and to convey some radio link control signalling messages. The access to RCHs is controlled with a contention window maintained by each MT. The number of RCH slots can be varied from frame to frame based on traffic load. (See [80] for a discussion on the efficiency of different adaptation mechanisms used in the RCH.) To get transmission opportunities the MTs have to provide resource requests (number of U-PDUs to be transmitted) to the AP scheduler. These are made either in the RCH, or the scheduler can directly acquire the resource requests from the MTs by a polling mechanism.

The data link layer also provides error control capabilities. To detect transmission error in protocol data units (PDU), binary Cyclic Redundancy Check (CRC) codes are used. The data is segmented to PDUs and protected by a 24-bit CRC code. Based on the results of error detection with the CRC codes the receiver notifies the transmitter with an Automatic Repeat Request (ARQ) feedback whether the transmitted PDUs have been successfully received or not. The erroneous PDUs are retransmitted. Next we discuss this mechanism.

3.3 Related Work and Their Application in HIPERLAN/2

A number of static ARQ schemes are known. We first discuss how three basic retransmission mechanisms, Stop-and-Wait ARQ, Go-back-N ARQ [86] and PRIME [71] can be applied to HIPERLAN/2. We then discuss our dynamic proposal in Section 3.4.

The ARQ scheme works in a frame by frame basis in HIPERLAN/2 as shown in Figure 3.2. The transmitter sends a number of data PDUs grouped together in a frame. The receiver makes one ARQ response within a frame, which contains the cumulative response to the received PDUs. The transmitter gets the ARQ feedback, and retransmits missing PDUs based on the ARQ information in the subsequent frames.

Stop-and-Wait ARQ means that the transmitter retransmits the last PDU until a positive acknowledgement arrives in response. The next PDU is transmitted only after the acknowledgement to the last transmitted PDU. This concept is illustrated in Figure 3.3. The transmitter and the receiver take turns to transmit the data and the acknowledgement packets.

The application of Stop-and-Wait ARQ is illustrated in Figure 3.4 in a sequence number diagram which plots the development of the ARQ protocol behaviour on a frame by frame basis. A packet of sequence number s is shown as a circle. If it is successfully received, an acknowledgement with a sequence number of $s + 1$ is sent back, shown as a triangle; otherwise an acknowledgement with a sequence number of s is sent.

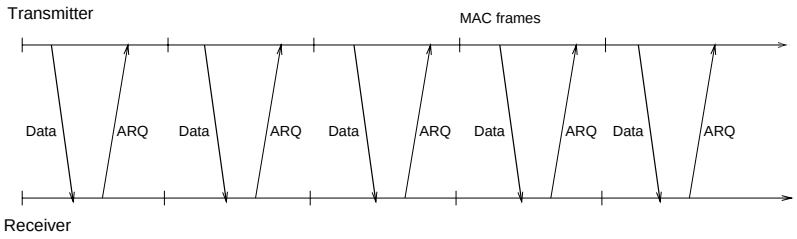


Figure 3.2: ARQ in the HIPERLAN/2 MAC frame structure (illustrated without ARQ processing delay).

The figure shows that this scheme is inefficient to apply to HIPERLAN/2 because the high capacity of the system is not utilized. The system has the capacity to send many PDUs per frame, but can only accommodate a single ARQ feedback message and therefore only one data PDU can be sent; this is why Stop-and-Wait cannot utilize the system resources.

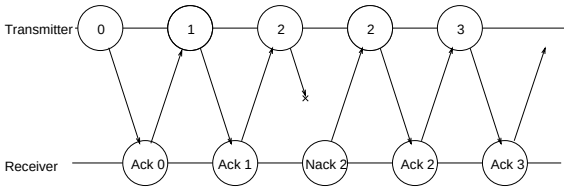


Figure 3.3: Stop-and-Wait ARQ example

With Go-back-N the receiver signals the first missing PDU, and the transmitter retransmits it (if there is a need for retransmission) as well as all consecutive PDUs. This concept is illustrated in Figure 3.5. The figure shows the transmitted PDU sequence numbers, and the receiver's cumulative acknowledgement, which gives the first missing PDU and hence implicitly acknowledges all previous PDUs. We can see that this scheme allows many data PDUs to be sent for each acknowledgement, but it may often happen that a PDU is retransmitted unnecessarily. Figure 3.6 shows the application of the protocol in the case of HIPERLAN/2. Again, one acknowledgement is sent per frame. The Go-back-N scheme provides a simple retransmission protocol, but a number of correctly received PDUs are also retransmitted, which decreases the performance.

To alleviate the inefficiency caused by retransmitting correctly received PDUs, a proposal called PRIME ARQ [71] (Partial selective Repeat superIMposEd on go-back-n) makes a compromise between Go-back-N and Selective Repeat ARQ. See Figure 3.7 for the illustration of this ARQ concept. In this proposal the receiver signals the first M missing PDU numbers, where M is a parameter of the protocol. The transmitter retransmits the indicated PDUs, and all consecutive PDUs above the highest of the M indicated PDU numbers. PRIME selectively retransmits only the missing PDUs if their number is less than M. (In contrast, the Go-Back-N scheme is a special case of PRIME when $M = 1$; if M goes to infinity, PRIME works as a selective repeat scheme, as described below.)

Figure 3.8 shows the application of this mechanism in HIPERLAN/2. One control PDU carries three missing PDU numbers in each frame. We can see that the PRIME protocol is much better in avoiding duplicate retransmissions than Go-back-N.

In Selective Repeat ARQ [86, 64], the receiver gives a detailed feedback on which PDUs

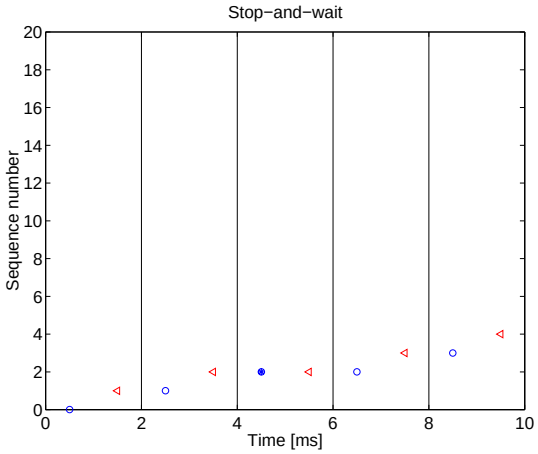


Figure 3.4: Stop-and-Wait ARQ in the HIPERLAN/2 system. Vertical lines mark frame boundaries; circles and triangles represent data and acknowledgement PDUs, respectively. A dark circle shows a data PDU that is lost (corrupted) in the transmission and hence must be retransmitted.

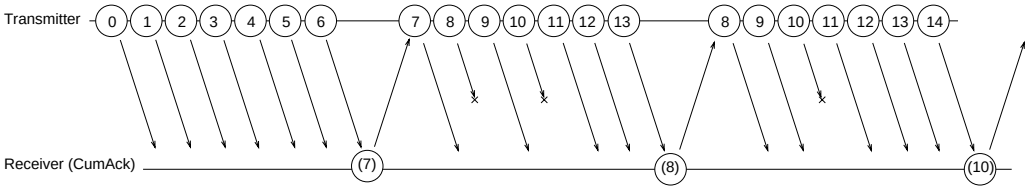


Figure 3.5: Go-back-N ARQ example

are received and which are not. The transmitter retransmits only those that are missing at the receiver. This concept is illustrated in Figure 3.9 where the receiver gives a cumulative acknowledgement (the sequence number of the first missing PDU) and a bitmap of the arrival status of the subsequent eight PDUs (one means received, zero means not received). This is one possible way of how selective repeat gives detailed feedback to the transmitter.

The protocol behaviour is shown in Figure 3.10 in the context of HIPERLAN/2. We illustrate the receiver feedback of the cumulative acknowledgement and the bitmap of the next eight sequence numbers. This is the most efficient mechanism in avoiding duplicate retransmissions, but at the cost of using the most feedback information.

Our motivation for developing a dynamic retransmission protocol is to allow for adaptation in the amount and contents of the ARQ feedback messages. This is needed in an environment with bursty traffic patterns and changing channel conditions.

An adaptation mechanism in the ARQ feedback complements other adaptive techniques known in the literature in the physical layer (adaptive channel selection, adaptive antennas, modulation mode selection, power control, processing gain adaptation, adaptive equalizers, etc.; see the references in [77, 15, 88]) and in the link layer are such as adaptive frame and packet sizes [59, 89, 90, 21, 54] and adaptive coding schemes in [42, 88, 59, 21, 23]. A very general approach to adaptation is presented in [7] where adaptation is achieved by re-programming the MAC layer.

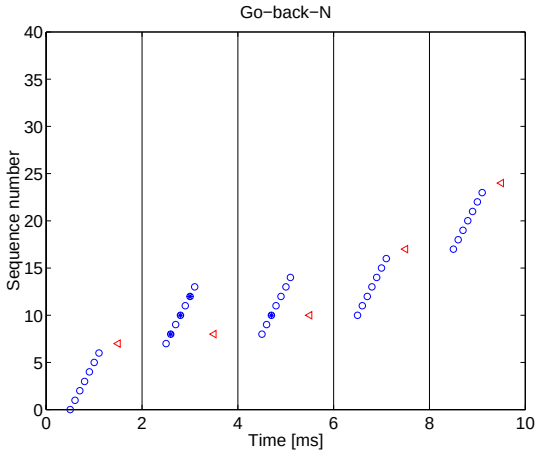


Figure 3.6: Go-back-N ARQ in the HIPERLAN/2 system (same notation as in Figure 3.4).

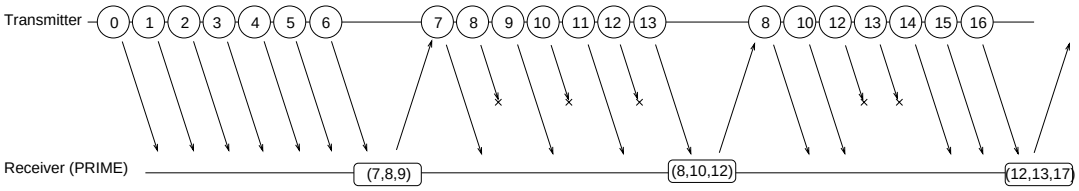


Figure 3.7: PRIME ARQ example

3.4 Dynamic Retransmission (ARQ) Protocol in HIPERLAN/2

As we have seen above, the HIPERLAN/2 system is dynamic in the sense that the data traffic changes on a frame by frame basis, as decided by the central scheduler. In addition, the amount of ARQ control messages also has to be decided: in HIPERLAN/2, they are also scheduled in the central scheduler. Furthermore, the contents of the ARQ can be dynamically adapted to the current conditions of the system. Another aspect that needs to be taken into account is that different implementations may impose extra processing delays. All these features make the design of the ARQ protocol difficult and motivate mechanisms.

To solve these problems, we have proposed a new version of selective repeat ARQ, called Selective Repeat ARQ with Partial Bitmaps (SRPB) [C5, 12]. The protocol allows the transmission and reception of PDUs in a very flexible way within the transmit and receive windows. Here we highlight the novelties of the protocol, see [41] for a full description.

The receiver maintains a bitmap corresponding to the reception status of each PDU in its receive window and uses this bitmap to give feedback to the transmitter. Based on this feedback, the transmitter retransmits the missing PDUs. The ARQ feedback in ARQ C-PDUs with the information fields are summarized in Table 3.2. In a single ARQ C-PDU, the receiver can signal three 8-bit portions of this bitmap, called bitmap blocks. Each bitmap block is identified by a bitmap number. This gives the receiver a great deal of flexibility in choosing which bitmap

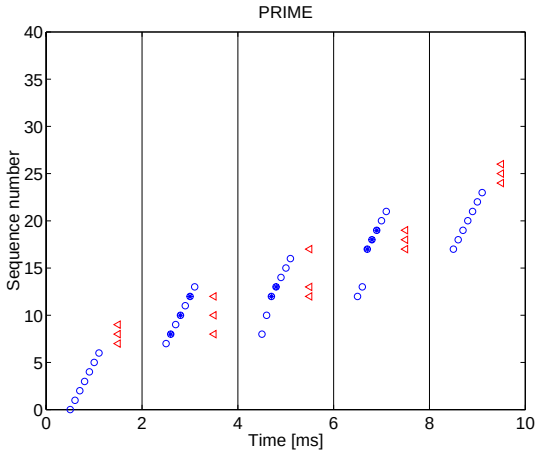


Figure 3.8: PRIME ARQ in the HIPERLAN/2 system (same notation as in Figure 3.4).

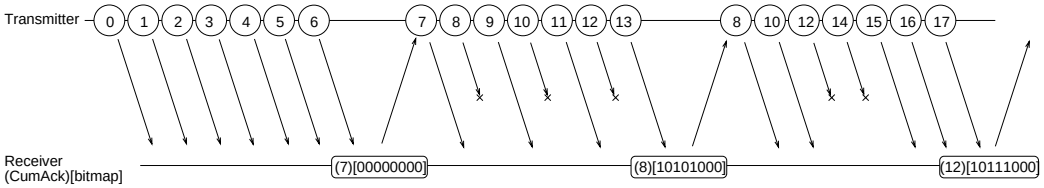


Figure 3.9: Selective Repeat ARQ example

blocks to signal so that the ARQ feedback is the most efficient.

In addition, the transmitter can cumulatively acknowledge earlier PDU receptions by the CAI (Cumulative Ack Indication) bit. When it is set, all PDUs before the first bitmap block have been correctly received. Cumulative acknowledgements help the transmitter become aware of the correctly received PDUs and advance its transmit window even if some of the ARQ C-PDUs are lost.

The standard allows flexibility in the design of the ARQ, since it only specifies the format of the ARQ messages, leaving room for optimizations in both the transmitter and the receiver. Regarding the receiver, it can use a number of strategies to generate the ARQ feedback. We provide here three simple mechanisms that the receiver can follow.

A In the first and simplest strategy, the receiver always signals the blocks continuously from the bottom of the window. This is the simplest strategy to follow, but it does not utilize the flexibility of the ARQ messages.

This strategy is illustrated in Figure 3.11 where the transmitted data PDUs and the acknowledgements are shown on a frame by frame basis. The acknowledgement contains three bitmap blocks and their block numbers. (Block number 0 corresponds to PDUs 0-7, block number 1 corresponds to PDUs 8-15, and so on.) In every frame, the transmitter sends eight PDUs in this example, and the receiver signals its reception status. The transmitter first retransmits the missing PDUs, and then continues with the transmission of new PDUs.

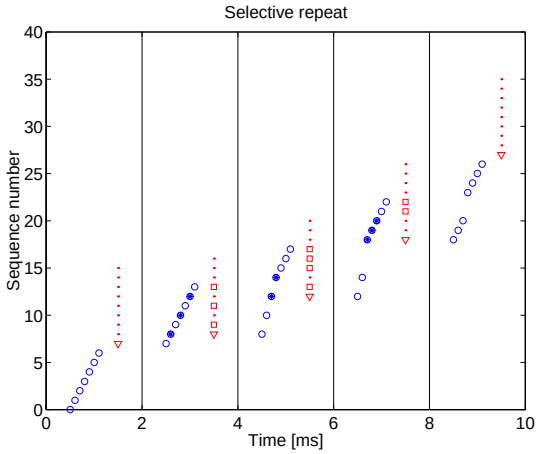


Figure 3.10: Selective repeat ARQ in the HIPERLAN/2 system (triangle means cumulative acknowledgement; dots represent missing, squares represent received PDUs; other notation as in Figure 3.4).

Field	Bits	Function
CAI	1	Cumulative Ack Indication
BMN 1	7	Bitmap Number 1
BMB 1	8	Bitmap Block 1
BMN 2	5	Bitmap Number 2
BMB 2	8	Bitmap Block 2
BMN 3	5	Bitmap Number 3
BMB 3	8	Bitmap Block 3
ABIR	1	ARQ Bandwidth Increase Request

Table 3.2: ARQ feedback fields

Because all the bitmap blocks are reported, sometimes a block with no errors is also signalled. This prevents the signalling of missing PDU 24 in Frame 4, which is why it is not retransmitted in Frame 5, but only in Frame 6.

B In the second strategy, the receiver signals the blocks beginning from the bottom of the window, but only those where there is an error. This makes the retransmission of errored PDUs faster. In the example of Figure 3.11, the bitmap block with all ones would not be signalled in this strategy.

We illustrate this strategy for the case when the round-trip time of the ARQ protocol is greater than one frame. The round-trip time is the minimum time necessary for a retransmission to arrive at the receiver after the negative acknowledgement is sent. The round-trip time of the ARQ protocol can be bigger than one frame because of processing delays in the transmitter or receiver.

See Figure 3.12 using strategy B in the case when there is a one frame delay in generating the ARQ feedback, hence the round-trip time is two frames. (Note that it would be also

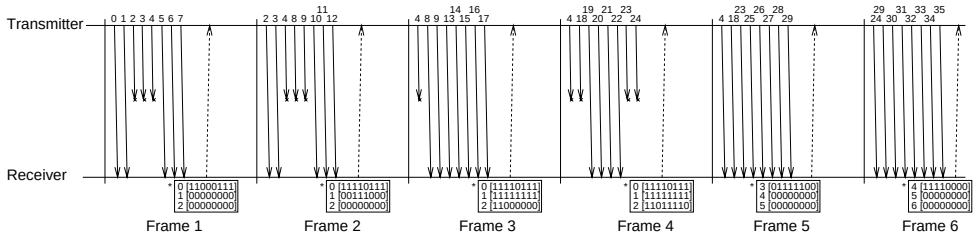


Figure 3.11: Receiver ARQ strategy A. The receiver chooses the first three bitmap blocks to signal in the ARQ control message. (The bitmap blocks are shown in brackets, preceded by the bitmap block numbers.)

possible that the transmitter introduces processing delays.) The transmitter is aware of the delay in the receiver, and retransmits only the PDUs that are missing and have been sent until the previous frame. (For this, the transmitter has to remember the last PDU that it sent in the previous frame.)

We can see in Frame 5 that bitmap block 1 is not reported since there are no errors in it. This saves ARQ signalling capacity.

On the other hand, the figure also illustrates a problem. Errors are reported twice before the first retransmission can arrive due to the delay in the ARQ generation. Since the transmitter retransmits negatively acknowledged PDUs, missing PDUs are retransmitted twice, even if the first retransmission is successful. This procedure wastes not only ARQ signalling capacity but also data transmission capacity. This motivates the use of strategy C.

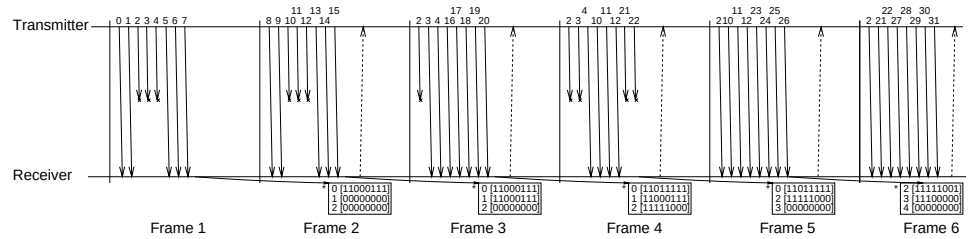


Figure 3.12: Receiver ARQ strategy B. The receiver chooses the first three bitmap blocks where there is a missing PDU. The receiver has a delay of one frame for generating the ARQ C-PDU.

C The third strategy is suited for the case when the round-trip time of the ARQ protocol is greater than one frame. In this strategy, the receiver signals the blocks from the bottom of the window, but only those where there is an error, and excluding those which have been signalled in the last round-trip time period (including this frame) and are unchanged. In addition, the receiver chooses the first bitmap block in the window at least once in a round-trip time to be able to cumulatively acknowledge all data below the window.

In Figure 3.13 we plot the use of strategy C. Note that in this case we do not always report the first bitmap block where there is an error. If we report the first block, the CAI (Cumulative Ack Indicator) is set, which is marked by an asterisk. In the figure, this is set in every second frame, as required by strategy C.

We can see that the missing PDUs that were reported twice in the previous example are now reported only once, and hence they are retransmitted only once. This saves both transmission capacity and ARQ signalling capacity. This is why in the six frames shown and with the same error pattern, strategy C can transmit up to PDU 36 while strategy B can transmit up to PDU 31 only. (Note however that it may still happen that a missing PDU is sometimes reported twice in this strategy, when a bitmap block is reported again after a change or when it is reported as the first block of the window, as in Frame 6.)

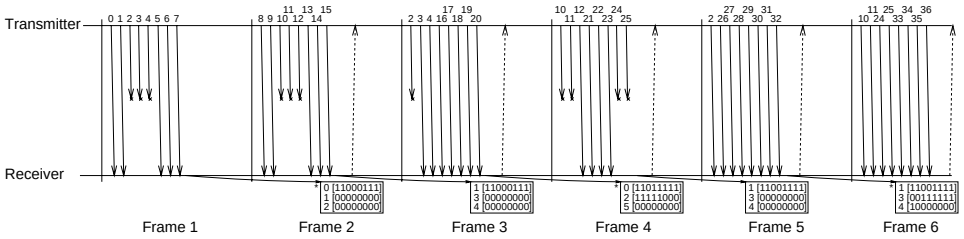


Figure 3.13: Receiver ARQ strategy C. The receiver excludes the bitmap blocks that have been signalled in the last round-trip time and are unchanged. The acknowledgements where the CAI (Cumulative Ack Indicator) is set are marked by an asterisk. At least one such ARQ C-PCU is generated in each round-trip time.

We have implemented strategy B in a packet-based simulator and compared it to PRIME ARQ in [C5]. Figure 3.14 shows one graph from the study. It shows the throughput performance of the SRPB and PRIME ARQ using the same set of constraints: a single cell was simulated on a bursty Markovian error model. As the offered traffic is increased, the throughput saturates at a higher value for the SRPB protocol. This can be attributed to the fact that the SRPB protocol is more efficient in retransmitting only the lost (corrupted) PDUs without retransmitting correctly received PDUs. For this reason, and because SRPB can incorporate a number of different strategies like A, B, and C, the HIPERLAN/2 standardization body decided on the use of the SRPB protocol.

Besides the flexibility of choosing the bitmap blocks to signal at the receiver, there is another feature that makes the ARQ protocol dynamic. The number of ARQ C-PDU feedback messages is controlled by the scheduler in the access point which may follow a dynamic scheme. Allocating more ARQ C-PDUs allows quicker retransmissions, but it also consumes more resources. The dynamic allocation of ARQ capacity is also supported by the ABIR (ARQ Bandwidth Increase Request) bit in the ARQ C-PDU. When this bit is set in an ARQ message sent by an MT, it signals to the AP scheduler that the MT would like to increase the number of ARQ C-PDUs.

This feature is illustrated in Figure 3.15. When the receiver can not signal all the errors, it sets ABIR to one, and the scheduler increases the allocated number of ARQ C-PDUs by one. If the ABIR is zero, the allocated number of ARQ C-PDUs is decreased by one, but at least one ARQ C-PDU is always provided.

The performance gain of using a dynamically adaptable number of ARQ C-PDUs is plotted in Figure 3.16. The adaptation of the number of allocated ARQ C-PDUs is done as shown in the previous figure. This allows a quick adaptation of the number of ARQ C-PDUs, which is more efficient than a static allocation. Dynamic allocation allows a much quicker signalling of missing PDUs when there are many errors, and therefore retransmissions are quicker.

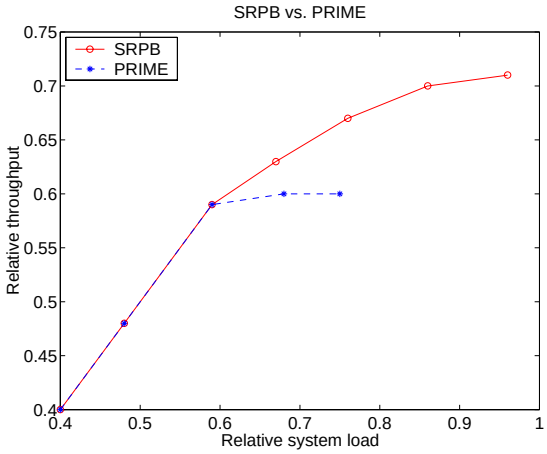


Figure 3.14: Throughput performance of SRPB vs. PRIME ARQ. Parameters: MAC frame size 2ms, ARQ window size 127, Mean packet error rate 10%, mean duration of GOOD state in the channel, mean duration of BAD state in the channel 2ms, parameter of PRIME $M = 3$. Load and throughput are plotted relative to the total system capacity.

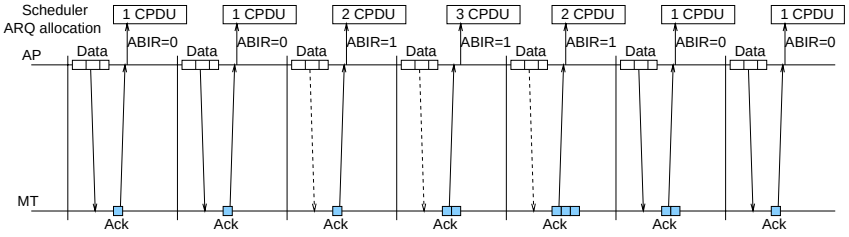


Figure 3.15: ABIR.

3.5 Summary

In HIPERLAN/2 the ARQ protocol has to provide feedback to a dynamically changing amount of traffic in such a way that it is sent in control PDUs that are scheduled separately from data traffic. Therefore, new mechanisms are needed to optimize the contents of the ARQ messages and schedule them. These mechanisms must work even if there are implementation-dependent delays in the end systems.

To satisfy these new demands, we have proposed a new dynamic selective repeat ARQ protocol, Selective Repeat Partial Bitmap (SRPB) ARQ. The protocol is dynamic in a number of ways. The receiver can dynamically choose the bitmap blocks that it signals in its ARQ message. We have proposed three simple schemes by which the receiver can choose which bitmap blocks to signal. In addition, the receiver can signal when it has too little capacity allocated for ARQ information, and based on this feedback, the scheduler dynamically assigns the amount of capacity for ARQ.

Simulation results show that SRPB gives higher performance than the PRIME protocol at high traffic loads because of the fact that it is more efficient in retransmitting only the corrupted

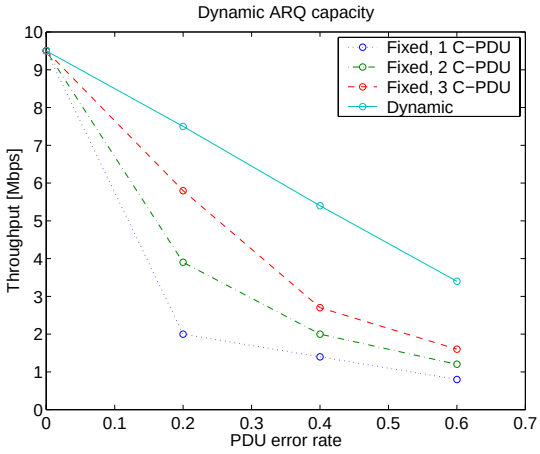


Figure 3.16: Performance of dynamic vs. fixed ARQ capacity allocation

packets. The simulation results also show that the dynamic adaptation of the capacity of the ARQ messages efficiently improves the system throughput.

Chapter 4

Fairness and Utilization Analysis of the Resource Allocation of a Wireless Base Station

4.1 Introduction

The question of fair resource allocation has been extensively studied in wired networks using the notion of fluid fair queuing. Using this concept, an ideally fair resource allocation can be defined among the flows sharing a common link [95, 5].

In fluid fair queuing, each flow i has a weight w_i. The total capacity is distributed in proportion to the weights among the flows that are backlogged at a given instant. Formally, for any time interval $[t_1, t_2]$ during which there is no change in the set of backlogged flows $B(t_1, t_2)$, the amount of allocation $W_i(t_1, t_2)$ provided to flow i is such that

$$\forall i, j \in B(t_1, t_2), \qquad \frac{W_i(t_1, t_2)}{w_i} = \frac{W_j(t_1, t_2)}{w_j} \tag{4.1}$$

This is known as the Generalized Processor Sharing (GPS) policy [73]. GPS has two important properties that motivates its use: it can provide a delay bound for the service as long as the flows are constrained by a leaky bucket, and it can ensure a fair allocation of resources for the backlogged flows regardless of whether or not their traffic is constrained. An important consequence of fair resource allocation is the separation property: a fraction of $w_i/\sum_j w_j$ of the resources is guaranteed to flow i even if all the flows are backlogged. If some of the flows become unbacklogged, the remaining resources are re-distributed among the backlogged flows. But the guaranteed amount of resources are never taken by other flows.

A number of packet fair queuing algorithms have been designed to approximate the ideal fluid fair queuing discipline in a practical algorithm. See [95] for an overview of these scheduling disciplines, including the most well-known example, Weighted Fair Queuing (WFQ) [73]. These algorithms differ in how accurately they approximate fluid fair queuing and in the complexity of their implementation. In Section 4.4 we will consider one such packet fair queuing algorithm in more detail, Start-time Fair Queuing (SFQ).

As we apply the fair queuing algorithms and results in a wireless environment, new problems arise. Because packets may be lost at the air interface, the amount of service allocated to a flow is not the same as the amount of service that the flow eventually gets. It implies that the fairness of fluid fair queuing as determined by Eq. 4.1 no longer holds. Since the channel error

characteristics may be location-dependent, the service that the flows get are not proportional to their weights.

To alleviate the unfairness problem, we introduce a compensation mechanism: we compensate for the resources lost due to errors on the air interface after these errors occur. The compensation mechanism increases the fairness among the flows, at the cost of reducing the separation property between the flows. We introduce a trade-off into the resource allocation problem: when we compensate the for the lost resources, we need to allocate more for the users with a lossy link, decreasing the overall channel utilization.

This trade-off is further complicated if the most prevailing transport protocol, TCP is used for transporting data over the wireless link and to dynamically make use of the available transmission resources, this can further modify the distribution of resources. As we discussed in Chapter 3, TCP reacts to packet losses as if they were caused by congestion, and decreases the sending rate. To hide this effect from TCP, we can use a dynamic link-layer ARQ protocol, as described in Chapter 3.

Our goal here is to investigate the interactions of these effects. We analyze the relationship between fairness and utilization when TCP is used as the transport protocol, a link-layer ARQ protocol is used to retransmit packets lost at the air interface, and a simple resource allocation architecture takes these losses into account and compensates for them.

This chapter is organized as follows. We begin with a brief discussion of related work in Section 4.2. Section 4.3 presents the architectural framework, Section 4.4 describes SFQ that will be extended with a compensation mechanism in Section 4.5. Section 4.6 analyzes the trade-off between fairness and utilization through simulations. Section 4.7 summarizes the results of this chapter.

4.2 Related Work

The question of how to design a link layer so that it performs well with TCP/IP protocols has been an active area of research and standardization. Showing the practical importance of the question, a working group of the IETF was formed called Performance Implications of Link layer Characteristics (PILC). Among the goals of the working group is to give advice to link layer designers on how to support TCP/IP protocols efficiently [26].

The potential performance problems of running the TCP/IP protocol over wireless links were first analyzed in detail in [14]. The authors identify the problem that due to the losses on the wireless link and losses during a handoff, IP packets may be lost. This is a problem because the most prevailing transport protocol, TCP [24, 85], was designed such that it interprets packet loss as a sign of congestion in the network and consequently decreases its window causing an unnecessary degradation in the throughput. The authors in [14] propose a fast retransmission scheme in the TCP end hosts to address the problem. Other authors also propose and analyze solutions that address this problem by modifications at the transport layer, see [1, 82, 93, 2, 13].

At the same time, there are also well-known methods to address the errors of the wireless physical link at the link layer [77, 86]. Forward Error Correction (FEC) uses redundant coding to be more resistant to bit errors. Automatic retransmission request protocol (ARQ) uses an error-detecting code and feedback to the transmitter to retransmit the errored data segments. These techniques can be applied at the link layer without requiring any knowledge about the transport and upper-layer protocols. As an alternative, [4] proposes the snoop protocol which performs local retransmissions based on transport layer knowledge over a wireless link without a native link-layer retransmission mechanism.

A comparision of the different proposals for improving TCP performance over a wireless link is found in [3]. The study concludes that link-layer retransmissions (including the snoop mechanism in [4]) are more effficient in reducing the effect of wireless transmission errors than transport

layer solutions. However, two specific problems may still arise with local retransmissions. If the local retransmission protocol does not guarantee in-order delivery of packets, then the transport layer retransmission mechanism of TCP may be invoked due to the presence of duplicate acknowledgements. (This problem is analyzed and an improvement using delayed duplicated acknowledgements is proposed in [91].) A second problem that may arise is that the local link-layer retransmission protocol and the transport layer end-to-end retransmission protocol may interfere, and an end-to-end spurious retransmission may occur while a local retransmission is still in progress.

The problem of spurious transport layer retransmissions in the case of local link-layer retransmissions at a wireless link was first studied in [18] and was regarded as a major performance bottleneck. However, [57] reconsiders the simulation results and concludes that a number of assumptions made in [18], such as a constant retransmission timeout of TCP, are unrealistic. Instead, [57] presents a new analysis of the problem and concludes that in cellular systems where the wireless link has a latency in the order of a few milliseconds, spurious retransmissions are very rare because of the conservative TCP-layer retransmission mechanism. This claim is also supported by measurements of simultaneous link layer and transport layer traces [60]. Experimental and simulation studies [21, 65] reinforce the finding that a local retransmission protocol at the wireless link makes a significant performance improvement without causing spurious transport-layer retransmissions. (Another approach to the potential problem of spurious retransmissions are addressed in [58] where some TCP extensions are proposed that reduce the likelihood of spurious retransmissions should there be a great variability in the round-trip-time due to wireless links.)

Based on these findings, [21] and [59] propose to use a persistent link-layer retransmission mechanism for reliable end-to-end flows. Such a link layer protocol hides the errors of the wireless link from TCP and avoids the triggering of congestion control mechanisms in response to non-congestion related losses. However, such a link-layer retransmission mechanism is only suitable for loss-sensitive end-to-end flows. In other cases, such as real time traffic, delivering the packets in a timely manner is more important than the avoidance of packet losses. If a packet does not reach its destination before its deadline, its local link-level retransmission is not only unneccessary, but also wasteful of resources. This is why [57] introduces the notion of flow-adaptive wireless links. A flow-adaptive link uses explicit QoS information in the IP header [27] to dynamically adapt physical and link layer error control schemes for each flow sharing the link. A flow-adaptive link uses persistent retransmissions for loss-sensitive TCP flow, while it avoids persistent retransmissions for delay-sensitive real-time flows. The PILC working group of the IETF came to similar recommendations. According to [30], a link that can separate between its flows should use a highly persistent retransmission policy for TCP flows and low persistancy for real-time flows.

Bhagwat et al. analyzes the interactions of link-layer scheduling and TCP performance in [6]. They consider the IEEE 802.11 WLAN architecture and show three scheduling schemes that work on a packet by packet basis based on feedback about the channel state. They identify the problem of head-of-line blocking when simple FIFO scheduling is used: in the case of erroneous transmission, a packet may be retransmitted several times due to the bursty nature of the wireless channel. This causes a waste of transmission resources. They show that a simple round-robin scheduling gives significant improvement in both utilization and fairness due to the fact that it avoids head-of-line blocking.

Gomez et al. present a framework for provisioning application and channel dependent quality of service [34]. The frameworks integrates three layers of adaptation. First, a channel predictor estimates the state of the channel based on previous transmission attempts. This is extended by a compensation mechanism (based on [6]) that gives compensation to flows with bad wireless link. Finally, an application-specific adaptation mechanism that takes into account how the application is able to adapt to the changes of the capacity of the wireless channel.

There has been a number of studies investigating the adaptation fair scheduling to the wireless environment (see [55, 56, 70, 76, 69, 22]) independently from the upper layer transport protocol and its interaction with the link layer scheduling. In this chapter we focus on the these interactions. Scheduling architectures will be considered in more detail in Chapter 5.

4.3 Resource Allocation Architecture

Figure 4.1 illustrates the considered architecture of resource allocation. This architecture is in harmony with the HIPERLAN/2 system introduced in Section 3.2 of Chapter 3, but the resource allocation architecture itself is general and can be applied in other systems as well.

We call each transmission/reception entity a user. Each connection of a MT corresponds to a user. The base station (BS, which is an Access Point, AP in the HIPERLAN/2 architecture) is represented by many users each belonging to a connection interfacing a MT. The BS contains the master scheduler that is responsible for the distribution of the capacity between the users. The output of the scheduler is the dynamic allocation of each fixed-length MAC frame. The unit of allocation is one radio PDU which is of fixed length.

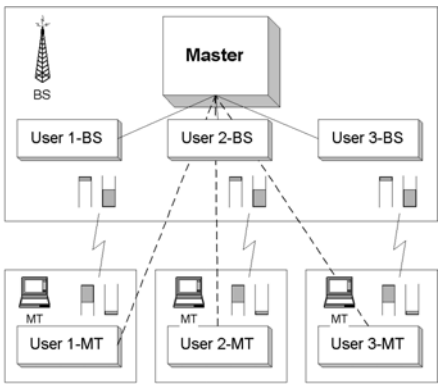


Figure 4.1: Resource allocation architecture

The master scheduler makes its resource allocation decisions based on the resource requests from the users. For the users within the BS the resource requests are just internal information within the base station. For the users in the MTs the resource requests are made to the master scheduler as control information. Resource requests specify the amount of data that the user has in its buffers to transmit.

The master scheduler runs the scheduling algorithm once for each frame. This algorithm takes the resource request from the users as its input and determines how much capacity is allocated to each user in the next frame in the unit of a radio PDU. Here we do not deal with the order of allocations within a frame. Our focus is the amount of allocations determined by the master scheduler for each frame.

4.4 Start-time Fair Queuing

Start-time Fair Queuing (SFQ) is a member of the family of packet fair queuing algorithms [95]. SFQ [35] greatly reduces the computational complexity of WFQ by avoiding the need to simulate the fluid server in real time. The virtual time variable used in SFQ is derived from the start tag of the packet in service. Another advantage of this method is that SFQ is applicable to variable

rate servers without a need to take the server rate into account in the virtual time computation [5]. The fairness, throughput and delay properties of SFQ are analyzed in [35].

Start-time Fair Queuing is defined as follows [35]:

1 Packet j of flow i is stamped with a start and finish time S_i^j and F_i^j, respectively, upon arrival at time A_i^j.

$$S_i^j = \max\{v(A_i^j), F_i^{j-1}\}, \tag{4.2}$$

$$F_i^j = S_i^j + \frac{L_i^j}{w_i} \quad \text{for } j \geq 1 \tag{4.3}$$

where F_i^0=0, w_i is the weight of flow i, L_i^j is the length of packet j of flow i, and $v(t)$ is the virtual time at time t.

2 The server virtual time is initially 0. During a busy period the server virtual time at time t, $v(t)$, is defined to be equal to the start tag of the packet in service at time t. At the end of a busy period, $v(t)$ is set to the maximum of finish tag assigned to any packets that have been serviced by time t.

3 Packets are served in increasing order of start tags; ties are broken arbitrarily.

We modify the first step in such a way that the start and finish tag are maintained for the first packet of a user only. (But even when the last packet of a user is served, its finish tag, denoted by F_i', must be maintained.)

1' A start and a finish tag, S_i and F_i, are associated with each *user*, corresponding to the virtual start and finish time of the packet at the head of the queue. When a new packet enters the head of the queue at time t (i.e., a packet has been served, or the user becomes backlogged), then the new values are computed from the old value of the finish time, F_i', as

$$S_i \leftarrow \max\{F_i', v(t)\}, \tag{4.4}$$

$$F_i \leftarrow S_i + \frac{L}{w_i} \tag{4.5}$$

where initially $F_i' = 0$, w_i is the weight of flow i, L is the length of all packets, and $v(t)$ is the virtual time at time t.

This modification does not have any influence on the SFQ itself, but it will make it easier to introduce compensation below.

4.5 Compensation for Lost Resources

We analyze the trade-off between fairness and utilization by introducing a very simple compensation mechanism. The master scheduler gives additional allocations to the users which have experienced errors. Errors become known to the master scheduler through the ARQ protocol after the errors occur.

To implement the compensation, we introduce the state variable *lag* for each user, denoted by n_i, to represent the amount of normalized service that the user should get in compensation. The constant β_l, $\beta_l \leq 1$ represents the amount of compensation for lost resources. $\beta_l = 1$ means that all of the lost capacity is compensated later, $\beta_l = 0$ means no compensation for lost capacity. As we will discuss, it is even possible to use negative compensation, $\beta_l < 0$.

The SFQ scheduling is extended as follows.

4 After each error of a packet of length L on the wireless channel for user i as reported by the ARQ protocol, its lag is incremented by β_l times the normalized service that was lost:

$$n_i \leftarrow n_i + \beta_l \frac{L}{w_i} \tag{4.6}$$

$$n_i \leftarrow \min\{n_i, n_{max}\}, \quad n_i \leftarrow \max\{n_i, -n_{max}\}. \tag{4.7}$$

(That is, the value of n_i is kept between the thresholds $[-n_{max}, n_{max}]$.)

5 Equation 4.5 is modified as follows:

$$l_c \leftarrow \min\{n_i, \beta_l \frac{L}{w_i}\}, \; l_c \leftarrow \max\{l_c, -\beta_l \frac{L}{w_i}\}, \tag{4.8}$$

$$F_i \leftarrow S_i + \frac{L}{w_i} - l_c, n_i \leftarrow n_i - l_c, \tag{4.9}$$

where l_c is the normalized compensation given during the service of the packet. Its value is $\beta_l \frac{L}{w_i}$ when n_i is positive, $-\beta_l \frac{L}{w_i}$ when n_i is negative.

This means that while a user is being compensated, the normalized service given when a single packet is served is artificially decreased by β_l times the normalized length of a packet. This can also be interpreted as increasing the weight of the user from w_i to $w_i/(1-\beta_l)$. The lag is decreased by the amount that was used up for the compensation.

The admission control criterion for a new flow in the case of SFQ is $\sum_{i \in \mathcal{U}} w_i \leq C$, where C is the total capacity of the link. This must be modified due to the compensation mechanism. As seen from the implementation of compensation a user is being compensated in such a way that its weight is in effect increased from w_i to $w_i/(1-\beta_l)$. The admission criterion for a new flow becomes, using the worst case scenario when all flows need compensation,

$$\frac{1}{1-\beta_l} \sum_{i \in \mathcal{U}} w_i \leq C. \tag{4.10}$$

If this criterion is met, then the amount of w_i capacity is guaranteed for user i. The fraction $\beta_l C$ of the total capacity is reserved for the compensation mechanism, and the remaining $(1-\beta_l)C$ is distributed between the users.

4.6 Simulation Results

We have implemented the following scenario in our packet-based simulator PlasmaSIM[39]: the bandwidth of a single base station is shared by 8 MTs numbered 1 through 8 whose weights are 1;1;2;2;4;4;8;8; (in fact, to a single MT two users correspond — one in the MT and one in the AP — that have equal weights). Note that here we are using weights only as a relative quantity, but as mentioned above, weights can also correspond to throughput guarantees. Each MT is receiving data through a single greedy TCP connection from a fixed host (greedy meaning a connection in which there is always data to send). Our TCP version is New-Reno [29], which can recover from losses more efficiently than the Reno variant without reaching timeout. The wireless link is the only bottleneck in our setup. There is a 10ms delay between the fixed host and the base station in both directions. For a summary of other simulation parameters, refer to table 4.1.

In this study we use three loss models to represent losses due to bit errors. The first model $\mathcal{M}_1$ has 5% independent (i.i.d.) loss probability for all PDUs; the second model $\mathcal{M}_2$ has 25% i.i.d. loss; while the third model $\mathcal{M}_3$ has 25% bursty loss as given by a Markov Modulated Bernoulli Process (MMBP): the loss probability is modulated by a two-state discrete-time Markov-chain.

MAC frame length	2ms
# PDU per frame	8
PDU length	48 bytes
System capacity	1.5 Mbit/sec
# MTs	8
Weights of MTs	1;1;2;2;4;4;8;8;
end-to-end RTT	20ms
Length of simulations	20sec
IP packet size	552 bytes
TCP/IP header	40 bytes
Buffer size at sender	100 PDU
Buffer size at receiver	100 PDU
Loss model $\mathcal{M}_1$	5% i.i.d. loss
Loss model $\mathcal{M}_2$	25% i.i.d. loss
Loss model $\mathcal{M}_3$	25% bursty loss, MMBP
Mean hold. times, $\mathcal{M}_3$	(5,17.5) [PDU transmit time]
Loss rate, MMBP of $\mathcal{M}_3$	(0.95, 0.05)

Table 4.1: Simulation constants

See table 4.1 for the parameters of the model. We use the same loss model in both uplink and downlink directions in all cases.

The ARQ protocol operating at the link layer of the simulator is a selective repeat mechanism that models the SRPB protocol described in Section 3.4 of Chapter 3. The protocol incorporates a discarding feature as well which allows the sender to give up further retransmission attempts after a certain number of trials. The knowledge of the number of lost PDUs in a frame are transferred implicitly by the ARQ protocol to the master scheduler in the base station so that it can perform compensation one frame later than the transmission errors actually occur.

Our performance metrics are based on the throughput values of the connections (there is a single connection per MT). We define four levels of throughput. The *allocated throughput* is the total transmission capacity allocated to a connection, i.e. the number of allocated PDUs per second times the bit length of a PDU. The *DLC-throughput* is the throughput above the ARQ protocol, i.e. the number of successfully transmitted PDUs per second times the bit length of a PDU. The DLC-throughput is less than the allocation throughput due to losses over the air interface. The *IP-throughput* is the throughput above the DLC layer, i.e. the number of successfully transmitted IP packets per second times the bit length of an IP packet. The IP-throughput is less than the DLC-throughput due to padding at the last PDU of a packet in segmentation. The *application-level throughput*, or *goodput*, is the throughput seen by applications, i.e. the number of bits transferred from the end-sender to the end-receiver per second. The goodput is smaller than IP-throughput because of the IP and TCP headers in each packet, the presence of acknowledgement packets in the reverse direction, and because TCP might allow unnecessary retransmissions.

We first illustrate the results of full compensation ($\beta_l = 1$). In our first scenario, we use model $\mathcal{M}_1$ (5% i.i.d. loss) for connections 1,3,5,7; and use model $\mathcal{M}_2$ (25% i.i.d. loss) for connections 2,4,6,8; and fully-reliable ARQ at the link layer. Figure 4.2 shows the throughput values for each connection normalized by the weights of connections (1,1,2,2,4,4,8,8 for connections 1 through 8 respectively). Figure 4.2 illustrates full compensation: by increasing the allocation for MTs with worse channels, the normalized DLC and IP-level throughputs are nearly identical, resulting in fair goodput allocation.

In order to achieve fairness in the goodput, as Figure 4.2 illustrates, we have to allocate more capacity to users with worse channel conditions. This implies a trade-off between the fairness of goodput seen by the users and the utilization of the system. To analyse this trade-off, we define the following metrics. *Utilization* is defined as the sum of the throughput of the connections divided by the system capacity, and is interpreted for each of the four levels of throughput. *Fairness* is characterised by the variability of the normalized throughputs of connections. Here we measure fairness by the coefficient of variation (standard deviation divided by the mean) of

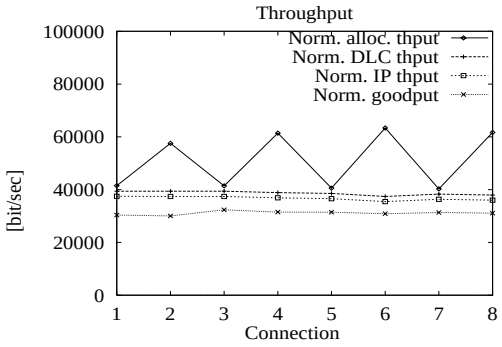


Figure 4.2: Throughput measures in case of $\mathcal{M}_1/\mathcal{M}_2$ models, fully reliable ARQ, full compensation. (The values for each connection are connected for better presentation.)

the normalized throughput values, and it is also interpreted for each of the 4 levels of throughput. (The smaller the fairness measure is, the more fair the allocation is. When the allocation of fair according to the fluid model of Eq. 4.1, this measure is zero.)

We first consider the same scenario with loss model $\mathcal{M}_1$ (5% i.i.d. loss) for odd-numbered and loss model $\mathcal{M}_2$ (25% i.i.d. loss) for even-numbered connections. Figure 4.3 shows the utilization and fairness measures as a function of the compensation constant.

The results show that it is possible to improve fairness of IP-level throughput and goodput by more than an order of magnitude. The goodput fairness improves monotonically with increasing compensation constant as a result of the compensation mechanism. This comes at the expense of a slight decrease in the utilization: goodput utilization drops by 1.8% as β_l increases from 0 to 1. By applying a negative compensation of $\beta_l = -2$, we can increase utilization slightly by 1.2%.

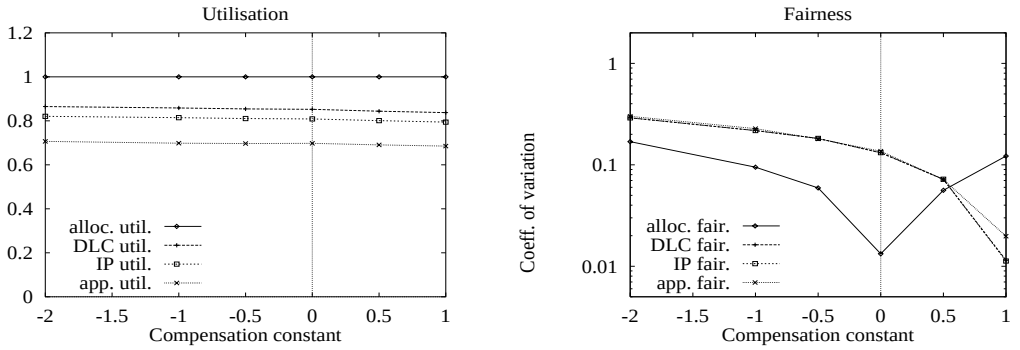


Figure 4.3: Utilization and fairness in the case of 25% i.i.d. loss for even-numbered connections.

The second scenario we consider is the same loss model $\mathcal{M}_1$ (5% i.i.d. loss) for odd-numbered connections but loss model $\mathcal{M}_3$ (25% bursty loss) for even-numbered connections. Figure 4.4 shows the results. We can observe a similar improvement of the fairness of goodput values, but this comes at the expense of significantly deteriorated utilization. (10.3% decrease in the goodput utilization as β_l goes from 0 to 1.) By applying negative compensation, we can improve goodput utilization by 4.8% at the expense of reduced fairness.

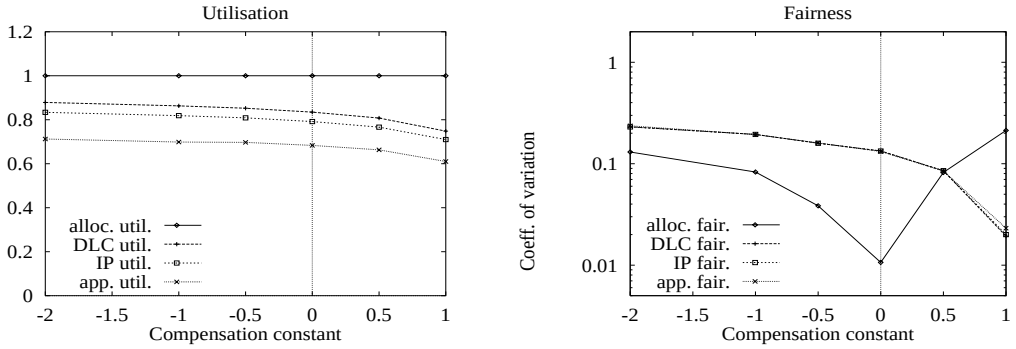


Figure 4.4: Utilization and fairness in the case of 25% bursty loss for even-numbered connections, fully reliable ARQ.

A fully reliable ARQ protocol in the link layer continues retransmissions indefinitely which may result in frequent losses in the case of bursty losses. We investigated the effect of limiting the number of retransmission attempts. The results for the limit of maximum eight transmission attempts of a given PDU are shown in Figure 4.5.

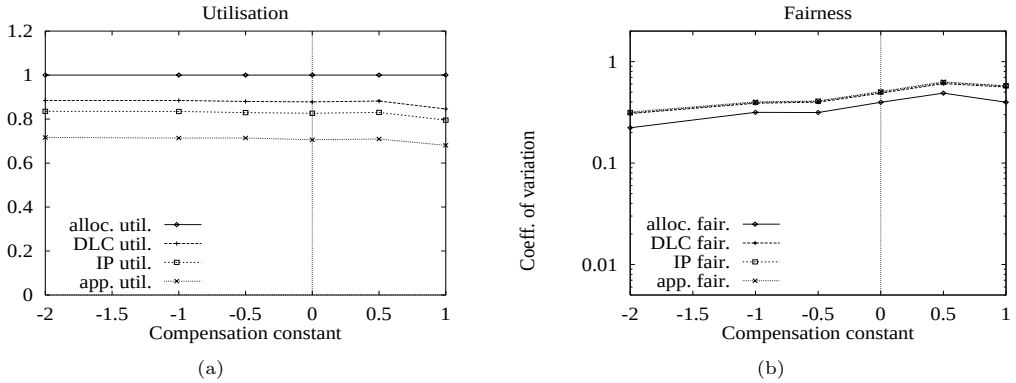


Figure 4.5: Utilization and fairness in the case of 25% bursty loss for even-numbered connections, semi-reliable ARQ.

The results show that using a semi-reliable ARQ the utilization can be slightly improved (for the $\beta_l = 0$ case it improved by 3.8%.) However, the allocation in this case becomes unfair and the fairness cannot be improved by increasing the compensation constant. Fairness in the case of semi-reliable ARQ protocol is determined by TCP's response to the losses not recovered by the ARQ protocol. It is interesting to note that fairness actually worsens in this case by increasing the compensation constant, which can be explained by the fact that additional capacity is allocated to a user following a loss, which in the case of bursty errors actually increases the probability of repeated losses and therefore IP packet drops.

Figure 4.6 shows the per connection throughput values ($\beta_l = 0$), implying that the relative decrease in the throughput of even-numbered connections ($\mathcal{M}_3$ loss model) is much higher for connections with higher weight, due to the fact that the throughput in this case is limited by the TCP congestion control mechanism.

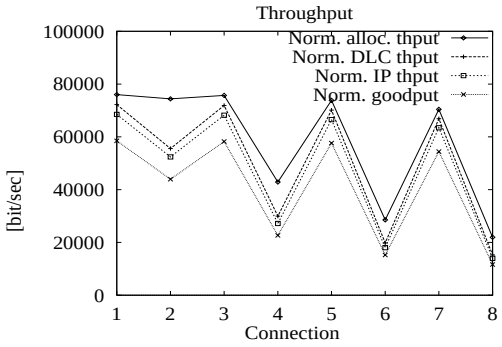


Figure 4.6: Throughput measures in case of $\mathcal{M}_1/\mathcal{M}_3$ models, semi-reliable ARQ, no compensation. (The values for each connection are connected only for easier presentation.)

4.7 Summary

We have presented a simple loss compensation scheme to analyze the trade-off between fairness and utilization of the resource allocation at a wireless base station. The performance evaluation of the proposed scheme has been carried out based on simulations that show the impact of TCP traffic and link-layer ARQ.

The results show that fairness of goodput values can be considerably improved by positive compensation at the expense of reduced system utilization. The reduction of utilization is slight in the case of independent channel losses, and increases with the burstiness of the loss model. Higher utilization is achieved by applying negative compensation or by using a semi-reliable ARQ at the link layer. The fairness of allocation can become considerably worse and can not be improved by the compensation mechanism in case of using semi-reliable ARQ.

Chapter 5

Distributed Scheme for the Resource Allocation of a Wireless Base Station

5.1 Introduction

The analysis in Chapter 4 used a simple compensation scheme in the master scheduler of the base station which controlled the trade-off between fairness and utilization. The scheduling scheme did not use any prediction or assumption about the errors on the wireless channel. Even without such knowledge, we have shown that the compensation scheme could significantly improve the fairness of the service received by the users.

We now investigate how to extend the simple compensation mechanism of the previous chapter to a practical scheme that takes into account the characteristics of the wireless channel as well. Wireless channel errors often have a bursty nature [6, 97, 77] and this may make it possible to predict the future state of the channel.

Even though it is plausible to assume the bursty nature of wireless channel errors and use this property in predictions, it is in general not possible to know the exact channel properties in advance. This is why our purpose is to design a resource allocation scheme that is independent of the specific assumptions on the channel. We do not assume bounds on the error rate, nor do we stick to a specific channel model.

We propose a novel resource allocation scheme that follows a distributed approach. We keep the simple architecture of the previous chapter. The master scheduler in the base station does not have any information about the current state of the wireless channel. Instead it compensates for channel errors after they occur. We extend the architecture by *allowing the users to defer their transmission to a later time when the channel is temporarily in a bad state*. Our approach is therefore decentralized in the sense that the master scheduler using a simple compensation mechanism is making the scheduling decision without any regard to the wireless channel state. Each user of the channel is responsible by itself for the estimation and prediction of its own channel, and can optimize when to transmit on its own. When the channel is expected to be bad for some time, the user can defer its transmission until the channel is expected to recover, based on the measured channel properties. This is encouraged (but not controlled) by the master scheduler as it allows the users to partially reclaim unused capacity in the future.

This approach offers several key advantages.

- We do not need to make any assumptions about the error characteristics of the channel

and the master scheduler does not need the prediction of the state of the link.

- Our scheme offers a modular implementation and greatly simplifies the master scheduler.
- It offers a decoupling of functionality: the users' estimation and prediction of the channel can be changed without modifying the master scheduler.

We will discuss the resource allocation scheme within the context of the resource allocation architecture presented in Section 4.3 of Chapter 4. Recall that this is a frame-based architecture. An important consequence is that the scheduler in the base station cannot get immediate feedback from the terminals on the success of the transmissions. Therefore the scheduling must be done such that packet transmission decisions are made at least one frame in advance, without exact knowledge of the momentary channel conditions.

The chapter is organized as follows. We review related work in Section 5.2. We discuss the design aspects of applying a fair scheduling scheme in our resource allocation architecture in Section 5.3. Our proposed master scheduling algorithm is described in Section 5.4. Section 5.5 shows how a user can optimize the channel usage for itself. Our simulation results are presented in Section 5.6, followed by an analytical approximation of the system performance in Section 5.7. Section 5.8 summarizes the chapter.

5.2 Related Work

The problems of applying the scheduling disciplines developed for fixed networks in a wireless context were first studied in [6]. They show that the FIFO packet scheduling employed in most commercial wireless LAN systems causes head of line blocking resulting in inefficient sharing of the bandwidth. To solve the problem, a new class of scheduling methods, referred to as channel state dependent packet (CSDP) scheduling methods were developed. The results show that it achieves significant improvements in utilization.

[55] investigated first the problems of applying a fair scheduling discipline in a wirelesss network. The paper identifies the problem that the service provided to a flow is not the same that the flow actually gets due to channel errors. They assume location dependent bursty errors, furthermore they assume that errors can be predicted in advance. With these assumptions they propose the Idealized Wireless Fair Queuing (IWFQ) algorithm which works by simulating a WFQ scheduler in the background, used to define the fairness model. Flows that receive more service than in the WFQ scheduler are said to be leading, flows that receive less are said to be lagging. The scheduler works by favouring channel access for lagging flows.

[70] and [56] propose two other adaptation of fair scheduling to wireless networks, CIF-Q (Channel-condition Independent Fair Queuing) and WFS (Wireless Fair Service), respectively. They apply a simulation of a fair scheduler in the background (Start-time Fair Queuing in CIF-Q, enhanced WFQ in WFS). However, compensation to lagging flows is different. While IWFQ stops giving service to leading flows to compensate lagging flows, in CIF-Q and WFS the service given to leading flows depends on the amount of lead. In CIF-Q, leading flows relinquish service as a linear function of their lead, while in WFS leading flows relinquish service as an exponential function of their lead.

[76] introduces the Server Based Fairness Approach (SBFA): instead of simulating a fair scheduler in the background, it uses the fair scheduling discipline itself to select the next flow for service. Besides the flows that carry regular traffic, virtual flows are introduced for compensation. When such a flow is selected by the scheduler for service, compensation is given to the corresponding flow. To achieve this, SBFA requires that we statically reserve a fraction of the capacity for the purpose of compensation. SBFA is very flexible in the sense that it can incorporate any fair scheduling algorithm into its framework.

[32] looks at the problem from a different perspective: rather than defining a new compensation mechanism based on fair queuing mechanisms, it defines an architecture for the capacity sharing of the link based on CBQ (Classed Based Queuing, [31]). It incorporates a modified version of CSDP scheduling [6] to alleviate the effect of channel errors.

[69] established a unified architecture in which all the proposals can be investigated and compares the algorithms mentioned above. By looking at the throughput, delay and fairness performance, [69] concludes that CIF-Q [70] and WFS [56] give the best results.

[22] accepts that location dependent errors may modify the fairness of user perceived throughput. The authors propose effort limited fair (ELF) scheduling as a compensation mechanism that ensures fairness up to a limited amount of errors, but at the same time it limits the amount of effort spent on a given flow. This scheme provides a flexible framework to administratively implement a variety of fairness and efficiency policies.

All of the above proposals use a centralized scheduler at the base station. The novelty of our architecture that we will discuss below is that we consider a decentralized architecture where the master scheduler in the base station does explicitly consider the characteristics of the wireless channel. Instead, we investigate how the master scheduler can encourage individual users to optimize the channel for their own.

5.3 Fair Queuing in the AP Master-scheduler

We first discuss how the resource allocation can accommodate a fair scheduling algorithm in the master scheduler. The scheduling process is detailed in Figure 5.1. The first row represents the implementation of the scheduling process. The input of the scheduler, namely the resource requests from users, are transmitted either in the random access channel at the end of frames (shown in the figure) or earlier in the frame as control information (not shown). Depending on the implementation the scheduling process can incur one or more frames of delay which we do not consider here. The output of the scheduler are the resource grants announced at the beginning of the new frame.

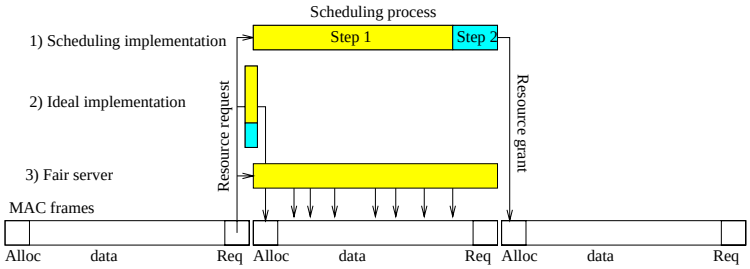


Figure 5.1: Scheduling process. The horizontal direction represents time, with the MAC frames shown at the bottom. Each frame begins with a broadcast of the frame structure announcement; frames end with a random access channel where the resource requests are sent.

When applying a fair queuing algorithm in the master scheduler, we can think of the scheduling process in two steps. Based on the resource requests from the users, a virtual server running a fair queuing algorithm serves the users in the first step, and the amount of total service is determined for each user for the next frame. This is the amount of allocation each user will get in the frame, but the actual ordering of allocation within the frame may be changed in the second step. Allocations for a given user can be grouped together to reduce overhead, and downlink and uplink transmissions can also be grouped together. The second row shows an ideal scheduler that performs these two steps immediately after the end of a frame to get the allocations for the

next frame. The ideal (and also the slower real) scheduler is based on a non-frame based fair server. It is shown in the third row, as it makes the scheduling decisions during the frame. We address only this step in this chapter, so it must be kept in mind that the actual transmissions may take place at a time of maximum one frame apart from when the server made the service.

It is possible to use any of the fair scheduling algorithms in the first step of the AP master scheduler [95]. A number of differences from wireline fair queuing must be noted however.

- The scheduling process has no immediate information about the arrival of packets or the immediate backlogged status of queues. All that the server can take into account is whether users are satisfied or unsatisfied as compared to the resource requests that users make on a frame by frame basis. This is why in this chapter we use the terms satisfied or unsatisfied, instead of backlogged or unbacklogged, for the status of the users in the scheduling process.

- The scheduler is only responsible for the amount of allocations to the users, and the users of the air interface themselves can decide how to utilize the allocated capacity for new transmissions and retransmissions. Since the scheduler is in no control of the individual packets, it is better to use per user state information in the scheduling process, rather than per packet information (as is traditional with the start and finish time tags in fair queuing algorithms).

- The computation of the system virtual time (i.e., the simulation of the fluid server) is made easier by the fact that users can become unsatisfied only at frame boundaries (i.e., when the scheduling for a new frame begins), which must be also packet service boundaries. This follows that the satisfied status of a user changes only after the service of a packet.

- The unsatisfied status of a user may change to satisfied even without the requested amount of service provided by the scheduler because the user may give up further transmission attempts of some packets at the expiration of a timer, or due to mobility. In addition, a user may decide to defer its transmission to a later frame as discussed below in Section 5.4 and Section 5.5. When the virtual fluid server is simulated, this requires modifications since the set of unsatisfied users can change in the past in the sense that virtual service was given to a user but the corresponding real service cannot be given later. To solve this problem, the real service to a user that is no longer unsatisfied can be given to another user as an extra service.

We will apply SFQ in our proposal, though other fair queuing schemes could also be applied taking into consideration the discussion above. The choice is motivated by the fact that SFQ is not sensitive to the changes of the satisfied status of the users, i.e., the computation of the virtual time or the selection of the next user does not have to be modified as users become satisfied and unsatisfied.

We adapt SFQ to the master-scheduler of the AP, as described in Chapter 4:

1 A start and a finish tag, S_i and F_i, are associated with each *user*, corresponding to the virtual start and finish time of the packet at the head of the queue. When a new packet enters the head of the queue at time t (i.e., a packet has been served, or the user becomes unsatisfied), then the new values are computed from the old value of the finish time, F_i', as

$$S_i \leftarrow \max\{F_i', v(t)\}, \tag{5.1}$$

$$F_i \leftarrow S_i + \frac{L}{w_i} \tag{5.2}$$

where initially $F_i' = 0$, w_i is the weight of flow i, L is the length of all packets, and $v(t)$ is the virtual time at time t.

2 The server virtual time is initially 0. During a busy period the server virtual time at time t, $v(t)$, is defined such that when a packet from user i is being served, the system virtual time becomes S_i, the start tag corresponding to the packet. When the system is idle the virtual time is increased linearly such that $dv(t)/dt = C/\sum_{i\in\mathcal{U}} w_i$ where $\mathcal{U}$ is the set of all users.

3 Packets are served in increasing order of start tags; ties are broken arbitrarily.

Recall that the start and finish tag computation was modified in so far as only the start and finish tag of the first packet of a user is maintained. F_i' is the finish tag of the last packet served.

5.4 Compensation for Unused Resources

We extend the compensation of lost resources of Section 4.5 of Chapter 4 with a new type of compensation: *compensation for unused resources*.

Unused resources correspond to the amount of service a user could have got while it did not request resources. By compensating unused resources later we give incentive to the users to monitor the state of their wireless links themselves and defer transmission when the link is temporarily in a bad condition.

We keep constant β_l of Chapter 4 to represents the amount of compensation for lost resources. Recall that $\beta_l = 1$ means that all of the lost capacity is compensated later, $\beta_l = 0$ means no compensation for lost capacity (we now use values, $0 \leq \beta_l \leq 1$). We introduce a new constant β_u to represent the amount of compensation for unused resources, $0 \leq \beta_u \leq 1$. Besides the constants β_l and β_u, we will also study the effects of the speed of compensation represented by γ, $0 \leq \gamma \leq 1$, which determines the increase of allocations when a user is being compensated. $\gamma = 0$ corresponds to no compensation, whereas $\gamma = 1$ corresponds to immediate compensation, where a user is compensated before any other users can get more allocations.

The SFQ scheduling algorithm is extended by giving compensation to users after errors have occurred (4), and when users miss service (5):

4 After each error of a packet of length L on the wireless channel for user i as reported by the ARQ protocol, its lag is incremented by β_l times the normalized service that was lost:

$$n_i \leftarrow \min\{n_i + \beta_l \frac{L}{w_i}, n_{max}\}. \tag{5.3}$$

5 When user i becomes unsatisfied at the beginning of a frame at time t after being satisfied, its lag is incremented by β_u times the normalized service that the user missed:

$$n_i \leftarrow \min\{n_i + \beta_u \max\{v(t) - F_i', 0\}, n_{max}\} \tag{5.4}$$

where $v(t)$ is the virtual time at time t, and F_i' is the finish time of user i before it became satisfied. Since $v(t)$ can be interpreted as the normalized fair amount of service that each user could have received up to time t, $v(t) - F_i'$ is the amount of normalized service that the user missed while it was satisfied.

6 Equation eq. (5.2) is modified as follows.

$$l_c \leftarrow \min\{n_i, \gamma \frac{L}{w_i}\}, \tag{5.5}$$

$$F_i \leftarrow S_i + \frac{L}{w_i} - l_c, \tag{5.6}$$

$$n_i \leftarrow n_i - l_c, \tag{5.7}$$

where l_c is the normalized compensation given during the service of the packet.

In this case compensation means that the normalized service when a packet is sent is compensated by γ times the normalized length of the packet. This corresponds to increasing the weight of the user from w_i to $w_i/(1-\gamma)$.

The virtual time computation was modified (4) in that for an idle server, the virtual time is incremented linearly so that it reflects the increase in the minimum amount of virtual service that each user could have been given if it were unsatisfied.

The amount of lag is maximized by n_{max}. Using this limit allows the compensation of lost and temporarily deferred transmissions but limit the amount of compensation for a users with little or no traffic for a long period of time.

Since we use now the constant γ for speed of compensation, this follows that the admission criterion for a new flow becomes $\frac{1}{1-\gamma}\sum_{i\in\mathcal{U}} w_i \leq C$.

5.5 User Behaviour

The previous section has presented the master scheduling algorithm that provides two kinds of compensation to the users: compensation for lost and unused resources. Each user can observe the quality of its own channel through measurements and can decide how much capacity to request. A user either makes a resource request according to the packets waiting for transmission in its buffers, or makes a resource request of zero, thereby relinquishing service to a later frame. Our purpose in this section is to arrive at a method for a user to decide when to relinquish service (i.e., make a resource request of 0). We assume that a user has knowledge of the scheduling algorithm and its constant parameters.

Relinquishing service in a given frame can be useful for the mobile because channel behaviour is typically positively correlated, so when a user observes bad channel, it is likely that the channel will continue to be bad for some time.

In the following subsections, we present a simple solution to this problem where the user builds a model of the channel estimating the parameters, which enables it to make a prediction of the future expected channel state and decide when to relinquish service.

5.5.1 Channel Model and Prediction

We use a simple AR(1) (autoregressive) model. This is a very simple model, yet powerful enough to capture the correlated nature of the channel. Note that the AR(1) model and the quantitative approximation must be regarded as heuristics since we have no way of getting any a priory information on the channel properties and its stationary nature.

We model the success rate in every frame, that is, the fraction of successfully received PDUs out of the transmitted PDUs in every frame. The distribution of errors within a frame is not modelled.

The AR(1) process (established independently for each user) is written as

$$v[t+1] = \rho v[t] + \sigma z, \tag{5.8}$$

$$s[t] = \pi + v[t] \tag{5.9}$$

where $v[t]$ is the actual AR(1) process, and $s[t]$ is the process of success rate. t is the frame counter (integer), z is a standard normally distributed random variable, ρ, σ, π are the parameters of the process. We make the simplification that the frames where the user is not scheduled are not regarded as part of the AR(1) process.

The expected value, variance and correlation of the process $s[t]$ are given as

$$\begin{aligned} \mathrm{E}\{s[t]\} &= \pi, & (5.10)\\ \mathrm{Var}\{s[t]\} &= \frac{\sigma^2}{1-\rho^2}, & (5.11)\\ \frac{\mathrm{Cov}\{(s[t], s[t+k])\}}{\mathrm{Var}\{s[t]\}} &= \rho^k & (5.12) \end{aligned}$$

We note that the process $s[t]$ could take on values outside the interval $[0, 1]$. However, we are going to use the model only to predict the expected value of the success rate in the future. According to Equation 5.17 in Subsection 5.5.1, it will be clear that the prediction cannot take on a value outside the interval $[0, 1]$ provided that π and the measured values of the process are within that interval.

We need to estimate the parameters of the model in such a way that as more and more measurements are available, the accuracy improves, on the other hand, the parameter estimation adapts to long-term, non-stationary changes in the channel behaviour. To achieve this, we use exponential moving averages in all the parameter estimations with weight ϕ.

Estimations are based on $s_i^*[t]$, success rate measured in the last frame. This is based on the assumption that the ARQ protocol is able to provide error characteristics for the last frame. π is estimated at frame t as

$$\pi[t] \leftarrow \pi[t-1]\phi + s_i^*[t](1-\phi) \tag{5.13}$$

Estimation of correlation is made indirectly through the estimation of second moment, δ, and first order mixed second moment (i.e., $\mathrm{E}\{s[t]s[t-1]\}$), ψ, of the process:

$$\delta[t] \leftarrow \delta[t-1]\phi + (s^*[t])^2(1-\phi),\ \psi[t] \leftarrow \psi[t-1]\phi + s^*[t]s^*[t-1](1-\phi) \tag{5.14}$$

The variance of the measured process is estimated as

$$\mu[t] = \delta[t] - (\pi[t])^2 \tag{5.15}$$

The following estimator can be shown to converge to the correlation if the measured process is AR(1):

$$\rho[t] = \frac{\psi[t] - \pi[t]\pi[t-1]}{\sqrt{(\delta[t-1] - (\pi[t-1])^2)(\delta[t] - (\pi[t])^2)}} \tag{5.16}$$

Given a current measurement of the success rate $s^*[t]$ and the estimations $\pi[t]$ and $\rho[t]$, the success rate for the frame $t+k$ can be predicted as

$$\hat{s}[t+k] = (s^*[t] - \pi[t])(\rho[t])^k + \pi[t] \tag{5.17}$$

5.5.2 User Decision

Based on the channel measurements and predictions, a user must decide whether to relinquish transmission in the next frame in the hope of more efficient transmission later. Our criterion aims at maximizing throughput for the user.

In order to be able to provide a decision function that is applicable by a user without any explicit information about other users or the future, we make several simplifying assumptions. We assume that compensation is not yet limited by the maximum lag; and we do not consider changes in the success rate during compensation for unused service; furthermore we approximate the service given to a user for a compensation of x to be $x\pi[t]$, that is, success rate during compensation is approximated by the estimated average success rate $\pi[t]$. Our decision will be

based on the expected service with compensation for one PDU in a frame.

If the user decides to transmit in the next frame, the expected service per packet in the frame is $L\hat{s}[t+1]$ (where L is the packet length), and the expected lost service is $L(1-\hat{s}[t+1])$. The expected compensation is $L(1-\hat{s}[t+1])\beta_l$ giving an expected service of $L(1-\hat{s}[t+1])\beta_l\pi[t]$ since we assume that the success rate during compensation for lost resources is $\pi[t]$. So the expected service per PDU is $L(\hat{s}[t+1]+(1-\hat{s}[t+1])\beta_l\pi[t])$. If the user defers transmission to a later frame $t+d$, the total service received instead of the allocation of a PDU in the current frame is less by the factor β_u since the service is being compensated for unused capacity. So the expected service per PDU is $L\beta_u(\hat{s}[t+d]+(1-\hat{s}[t+d])\beta_l\pi[t])$. Transmission is relinquished in the current slot if the expected service per PDU is increased:

$$L(\hat{s}[t+1]+(1-\hat{s}[t+1])\beta_l\pi[t]) < L\beta_u(\hat{s}[t+d]+(1-\hat{s}[t+d])\beta_l\pi[t]) \tag{5.18}$$

which gives

$$\frac{\hat{s}[t+1]}{\beta_u}+\frac{\frac{1}{\beta_u}-1}{\frac{1}{\beta_l\pi[t]}-1} < \hat{s}[t+d] \tag{5.19}$$

Deferring transmission in the current frame is useful when the channel model suggests that at a later frame $t+d$ the expected success rate fulfils the above equation.

We introduce a threshold-based decision criterion for a user using an upper threshold θ_1 and a lower threshold θ_2 on the expected success rate for the next frame $\hat{s}[t+1]$. The criterion is slightly different for a greedy and a non-greedy user. We can differentiate between a greedy and non-greedy user by a threshold on the buffer occupancy.)

The upper threshold determines the sensitivity of the algorithm and it is specified through the constant ω which gives the thresholds relative position with respect to the average success rate and its standard deviation:

$$\theta_1 = \pi[t]-\omega\sqrt{\mu[t]} \tag{5.20}$$

The lower threshold is determined in such a way that if the expected success rate in the next frame falls below the lower threshold and the user waits until the expected success rate reaches the upper threshold, then the user is expected to receive more service compared to not deferring transmission. This follows that the relationship between θ_1 and θ_2 can be obtained by applying eq. (5.19):

$$\theta_2 = \beta_u\left(\theta_1-\frac{\frac{1}{\beta_u}-1}{\frac{1}{\beta_l\pi[t]}-1}\right). \tag{5.21}$$

The rule for a greedy user is that we begin deferring transmission (i.e., requesting zero capacity) when the expected success rate falls below θ_2, and re-start when the expected success rate is above θ_1. In other words, we use a hysteresis with two thresholds.

For a non-greedy user, the rule may be different. In that case, the user has less data than the available capacity. In this case, if we defer the transmission to a later time, this does not mean that current transmission capacity is lost, since the user needs less capacity than available anyway. For a non-greedy user, we simply require that the expected success rate be at least θ_1.

Figure 5.2 shows the user decision rule. The outcome is either $R=0$ in which case the user relinquishes transmission in the current frame, or $R=1$ when the user transmits in the current frame. H is a state variable associated with the hysteresis, initially it is OFF.

In addition to the presented algorithm, we applied the following extra criteria: a user is not allowed to relinquish transmission in more than D_{max} consecutive frames, where D_{max} was set to 10. This ensures that the user periodically samples the state of the channel, and is not allowed to wait forever. Additionally, relinquishing transmission is switched off at the start of a transmission

$$
\begin{array}{llll}
H == ON & \hat{s}[t+1] < \theta_1 & R \leftarrow 0 & H \leftarrow OFF; R \leftarrow 1 \\
H == OFF & \hat{s}[t+1] \leq \theta_2 & H \leftarrow ON; R \leftarrow 0 & R \leftarrow 1 \\
 & \hat{s}[t+1] < \theta_1 & R \leftarrow 0 & R \leftarrow 1
\end{array}
$$

Figure 5.2: User decision rule

until the estimated parameters reflect measured values.

5.5.3 Adaptive Sensitivity Setting

The previous subsection described how the thresholds θ_1 and θ_2 can be computed given ω, which determines sensitivity of the thresholds. A high value of ω results in lower thresholds and therefore less frequent relinquishing of service. A low value of ω on the other hand causes the user to relinquish service more often.

To reach the highest throughput and most efficient resource utilization, a user should relinquish service as often as possible so that during transmission, the channel behaviour is as good as possible. However, the speed of compensation in the master-scheduler (determined by the parameter γ) poses an upper limit on the frequency of relinquishing service: the average rate of necessary compensation generated by the user should not exceed the rate of compensation that the scheduler can provide.

The optimal value of ω depends on the speed of compensation, γ, but also on the behaviour of other users, whether they request resources or not and how much compensation they receive. This is why we have chosen to implement an adaptive algorithm for setting ω based on feedback from the scheduler in the AP. This adaptive algorithm can be regarded as optional in the scheme: without it, a value for ω can be set as a constant.

The algorithm works as follows. In the broadcast channel of each frame, the AP gives a one-bit feedback to all the users which is set when the user has nonzero lag. Each user monitors this bit and computes an exponential moving average (denoted by f_i) for the ratio of frames where the user has nonzero lag at the scheduler. When f_i is too small this means that the scheduler can still provide more compensation. In this case the user decreases ω in order to get more compensation for unused resources, so that user is active when the channel is better. When f_i is too large this means that there is a danger that the lag is too high and reach its upper limit, i.e. the scheduler can not provide the required amount of compensation. In this case ω is increased.

Section 5.6 will show simulation results using adaptive threshold (ω) computation. In the simulated algorithm, the target for f_i is to be between 0.7 and 0.9, so ω is decreased when $f_i < 0.7$ and increased when $f_i > 0.9$. But ω must remain always within $\omega_{min} = 0.2$ and $\omega_{max} = 2$.

5.6 Simulation-based Performance Analysis

We have implemented our scheme in a packet-level simulation environment. The simulated architecture conforms to the resource allocation architecture presented in Section 4.3 of Chapter 4 with frame based MAC protocol with a frame length of 2 ms (as in the HIPERLAN/2 system [37]). In the first set of our simulations, the capacity of the system was 8 PDUs/frame where a PDU can carry 48 bytes, giving a total system capacity of 1.5Mbps. This capacity is shared by two users, each having an average PDU loss rate of 25%, but independent channel error models. User 1 has independently distributed losses, while User 2 has bursty losses according to an ON-OFF Markovian model. The average OFF period is 100 PDU transmission times, the loss rate is 95 % in the OFF period, 5% in the ON period. Both users are greedy which means that they transmit as many PDUs as possible.

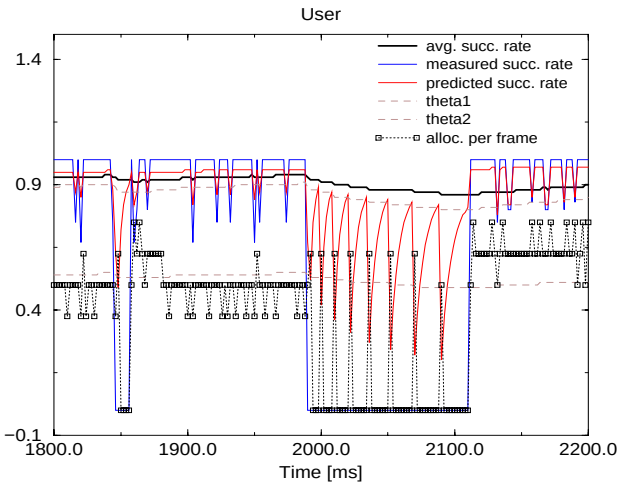


Figure 5.3: User behaviour when the channel has bursty errors.

The most widely used transport protocol, TCP, can be regarded as an approximation to such a greedy user (we do not deal with TCP layer and link layer interactions here, refer to Chapter 4 on this issue). Our performance metrics will be the achieved throughputs of the users, overall system utilization and fairness between users.

5.6.1 Example Trace of User Behaviour

Figure 5.3 shows a trace from a simulation for User 2, having bursty losses. The figure plots the time evolution of a number of parameters computed each frame. The measured success rate is the ratio of the successful PDU deliveries to all PDU transmissions in the last frame. The average and predicted success rates are computed as described in Subsection 5.5.1. The thresholds θ_1 and θ_2 (Subsection 5.5.2) are also plotted. The user's decision and the behaviour of the scheduler is seen on the amount of allocation per frame. In the figure, this gives the number of PDUs allocated in the frame to the user as a fraction of the total number of PDUs per frame (8 in our setup). When the user decides to relinquish service in a frame, this is seen by the lack of allocation per frame.

The simulation illustrates the user behaviour and compensation (we used the parameters $\gamma = 0.5, \beta_l = 0.5, \beta_u = 0.8$ here). The user begins to relinquish service when the predicted success rate falls below the lower threshold θ_2, and begins requesting again when the success rates reaches the upper threshold θ_1. In the figure we can see two periods when the user relinquishes service, one at 1850 ms, the other from 1980 ms to 2120 ms. The latter is a long OFF period in which the user re-starts several times. Note the increased amount of allocation after a user relinquished its service as a result of compensation for unused capacity in the scheduler.

5.6.2 Dependence on Parameter Settings

Here we investigate the dependence of the user performance metrics on the parameters of the scheduling scheme. Figure 5.4 shows the effect of ω on the throughput of connections and system utilization. In this simulation there was full compensation for unused capacity ($\beta_u = 1$), no compensation for lost capacity ($\beta_l = 0$) and the speed of compensation was $\gamma = 0.5$. The figure shows the following throughput quantities normalized by the system capacity: the total allocation

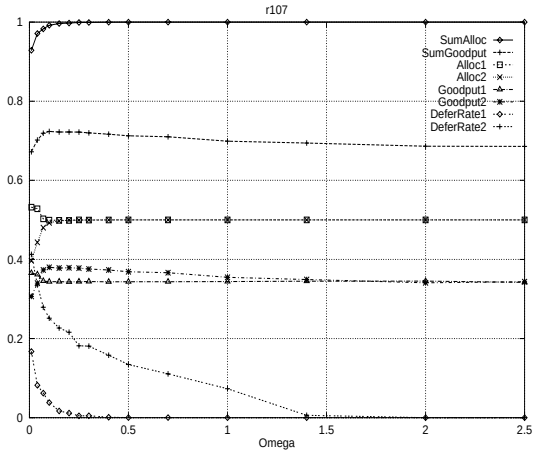


Figure 5.4: Effect of ω, $\gamma = 0.5$, $\beta_u = 1$, $\beta_l = 0$.

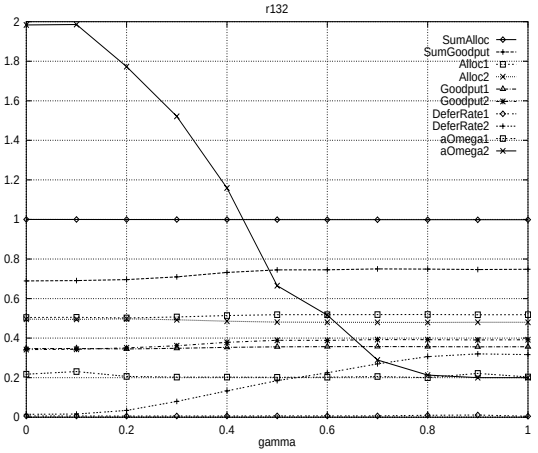


Figure 5.5: Effect of γ, $\beta_u = 1$, $\beta_l = 0.5$.

for both connections (SumAlloc), the total goodput of the two connections (SumGoodput), the allocated capacity from the two connections (Alloc1 and Alloc2) and the goodput (successfully carried traffic) for the two connections. The graphs DeferRate1 and DeferRate2 show the fraction of frames where User 1 and 2 decided to relinquished transmission.

The figure clearly shows a maximum for the total goodput and the goodput for User 2 (the one with bursty channel loss model) at near $\omega = 0.15$. User 1 can not achieve any improvement since its channel losses are independent. For higher ω, the gain of User 2 is less because the user relinquishes transmission less frequently, therefore it does not avoid all the bad channel states that it possibly could. This clearly shows the advantage of using the compensation method for unused capacity. For smaller ω, the compensation mechanism becomes saturated: the user relinquishes more service that the scheduler can compensate. As a result, the allocation to User 2 is decreased, and this results in goodput decrease.

Figure 5.4 illustrates that there exists an optimal value of ω motivating the use of the adaptive ω calculation approach described in Subsection 5.5.3.

Using that algorithm to set ω adaptively, Figure 5.5 shows the effect of changing the speed of compensation, γ, while $\beta_u = 1$ and $\beta_l = 0.5$. (The average value of ω for the users are shown by

aOmega1 and aOmega2.) As the speed of compensation increases, there is time for more frequent relinquishing of service for User 2 (increase of deferRate2). This is achieved through the decrease of ω. Note that as γ increases from 0 to about 0.6, there is significant increase in the total system goodput as well as the goodput of User 2 and also User 1. The increase of goodput for User 2 is explained by its better utilization of the allocated capacity, while the slight increase of goodput for User 1 is explained by the increase of allocation provided to it due to the compensation for lost capacity.

5.6.3 Performance of User Behaviour

We now turn our attention to a single user. The difficulty here is that the performance of the scheduling algorithm and user behaviour are closely coupled. Nevertheless it is possible to investigate the performance of user behaviour separately.

The net effect of the user relinquishing service is that, by making use of the positive correlation in the channel, the user can avoid transmitting when the channel is temporarily bad, and therefore the success rate of transmission is increased. This can be quantified by the plot of the user's observed success rate (π) versus the sensitivity of the algorithm as controlled by the parameter ω, which determines the frequency of relinquishing service.

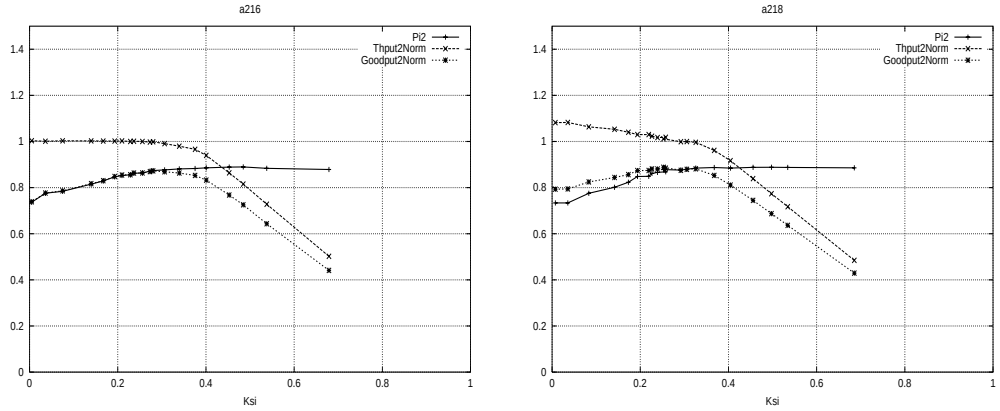
In the subsequent results, we use the following parameters simulation: the capacity of the system is 32 PDUs/frame where a PDU can carry 48 bytes, giving a total system capacity of 6Mbps. This capacity is shared by several users. Users 1,3,5,... have a "good" channel with a PDU loss rate of 5% and independent losses. Users 2,4,6,... have a "bad" channel with bursty losses according to an ON-OFF Markovian model with and average loss rate of 25%. The average OFF period is 100 PDU transmission times, the loss rate is 95 % in the OFF period, 5% in the ON period. All users are greedy, which means that they transmit as many PDUs as possible.

Figure 5.6 shows the simulated performance of the user algorithm as a function of ξ, the frequency of relinquishing service. This means the fraction of frames when the user decides to relinquish service. In this simulation there were 4 users, two with "good" channel and two with "bad" channel. The figure shows the success rate observed by the second user as it changes the sensitivity of its algorithm. (The other users employ automatic sensitivity setting.) It also shows the amount of allocation and amount of successfully carried traffic (goodput) normalized by the fair share of the user (i.e., 1/4 of the total capacity). In the simulations, $\gamma = 0.5$, β_u=1, and (a) $\beta_l = 0$ (b) $\beta_l = 0.5$.

The amount of improvement for the channel success rate shows the performance of the algorithm. (Note that these improvements are dependent on the parameter settings, the number of users, and channel properties.) We can observe that the algorithm works successfully by improving the success rate from 0.75 to 0.88.

The amount of allocation remains close to the fair share up to about $\xi = 0.35$, and then begins to decline. This decline is due to the same saturation effect as that seen in Figure 5.4: above the saturation point, the master scheduler can not provide the compensation for unused resources during the time when the user is requesting service. Below the saturation point, the scheduler provides full compensation for the unused resources, as illustrated in Figure 5.6 (a) by the horizontal line at 1. Figure (b) on the other hand shows that this user is allocated even more resources than its fair share below the saturation point. This is because in this simulation, there is compensation for lost resources ($\beta_l = 0.5$) and the more errors there are on the channel, the more extra allocation the user receives.

The declining curve of the amount of allocation to the user multiplied by the increasing curve of improving transmission success rate gives us the change of user goodput (successfully carried traffic). The user is looking for the maximum of this curve in order to optimize its throughput locally. We call it "locally greedy" user behaviour. Indeed, the adaptive sensitivity setting for this user finds the point at $\xi = 0.25$, which gives a nearly optimal local throughput performance.



(a) (b)

Figure 5.6: Performance of user algorithm as a function of ξ, the frequency of relinquishing service. $\gamma = 0.5$, β_u=1, (a) $\beta_l = 0$ (b) $\beta_l = 0.5$.

5.7 Analytical Approximation of System Performance

The simulation results above have validated the proposed architecture. On the other hand, based on purely simulations, it is hard to predict system performance in a quantitative fashion. For this reason, we propose here an abstraction which, although not as accurate as simulations, provides an analytical formulation of system utilization and fairness.

Besides the locally greedy user behaviour described in the previous subsection (Subsection 5.6.3, illustrated in Figure 5.6), we introduce here two simple other behaviours as well for analytical purposes. The first is the "no algorithm" case, that is, when the user does not relinquish service. This simply corresponds to the $\xi = 0$ case.

The second is the ideal algorithm case which is interpreted as follows. In the simulation example, the user algorithm has been able to increase the success rate from 0.75 to 0.88. We can say that an upper limit on the achievable success rate is 0.95, since this is the success rate in the ON state in our actual channel model. So the quicker and close the success rate approaches 0.95 as ξ is increased from 0 to 1, the better the algorithm is. The limiting case is the one in which the success rate reaches 0.95 as ξ grows from 0 to an infinitesimal small value. This takes us to the ideal limiting case, for which $\xi = 0$ and the success rate is 0.95.

In general, we say that an ideal algorithm is one in which for $\xi = 0$, the success rate is no smaller than what can be achieved by any $\xi > 0$. Such an algorithm is naturally impossible to implement or even approach, but it provides a valuable abstraction for the analysis of the system.

Figure 5.7 marks the "ideal" and "no algorithm" cases in a schematic figure. It also plots the marks the "locally greedy" point, where the goodput of the user is maximal.

The two abstract cases introduced above, the "no algorithm" and "ideal algorithm" cases, allow us to derive system performance metrics. In both of these cases, $\xi = 0$, i.e., the user does not give up service by itself. This follows that the total system capacity is distributed among the users based on their weights. However, we have to take into account the compensation mechanism for lost capacity.

Let us denote by Π_i the transmission success rate of user i. The normalized service without compensation to the user is L/w_i for one PDU. With probability $1 - \Pi_i$, the transmission is unsuccessful, in which case the lag of the user is increased by $\beta_l L/w_i$.

In order to derive the effect of compensation for lost capacity, we have to find the set of parameters where the compensation is saturated. This happens when the amount of lag generated

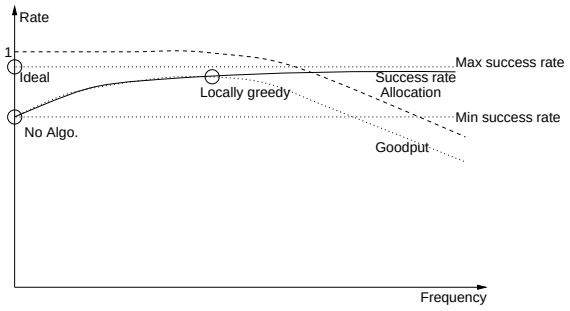


Figure 5.7: The "ideal" and "no algorithm" cases, as well as the "locally greedy" algorithm. The schematic figure plots the success rate, rate of allocations and rate of goodput as a function of the frequency of relinquishing service, as in Figure 5.6

is at least as much as the maximum amount of compensation that the scheduler can give to the user. The expected increase of the lag for one PDU is $(1 - \Pi_i)\beta_l L/w_i$, and the maximum compensation is given when the user gets compensation during the service of all PDUs, in which case the compensation if $\gamma L/w_i$ per PDU. This follows that compensation is saturated when $\gamma < (1 - \Pi_i)\beta_l$.

In this case, the normalized service with compensation for one PDU is $L/w_i - \gamma L/w_i$. If the compensation is not saturated, then the compensation equals the amount of lag generated, so normalized service for one PDU is $\Pi_i L/w_i + (1-\Pi_i)(1-\beta_l)L/w_i$. Taking the two cases together, the normalized service to the user is compensated in such a way that it can be modelled by the following modified weight of user i:

$$w_i' = \frac{1}{1 - \min\{\gamma, \beta_l(1 - \Pi_i)\}} w_i. \tag{5.22}$$

The total system capacity, C, is distributed among the users according to equation 5.22. Using $W' = \sum_i w_i'$, the allocation and goodput to user i (assuming greedy users) are

$$a_i = \frac{w_i'}{W'} C \text{ and } g_i = \Pi_i a_i. \tag{5.23}$$

The total system performance is characterized by the following metrics: the system utilization, defined by $U = \sum_i g_i/C$, the goodput fairness $V_g = \sigma\{g_i/w_i\}/E\{g_i/w_i\}$, and allocation fairness $V_a = \sigma\{a_i/w_i\}/E\{a_i/w_i\}$, that is, the coefficient of variation of normalized goodput and allocation values, as used also in Chapter 4. The higher the coefficient of variation, the worse the fairness is.

For a specific channel, we can use the following success ratios: $\Pi_{i,avg}$ for the average success rate of the channel, and $\Pi_{i,max}$ for the maximum success rate that can be achieved by any user behaviour. Such an upper limit can be interpreted, for example, for a Markovian channel by the success rate in the best state. By using the success rates $\Pi_{i,avg}$ and $\Pi_{i,max}$ for Π_i, the utilization and fairness metrics can be derived for the no algorithm and ideal algorithm cases.

Figure 5.8 (a) and (b) show these metrics for the case of four users, $\beta_l = 0.5$ and $\beta_l = 1$, respectively, and the same parameters as in Subsection 5.6.3. The figures show the two abstract cases (no algorithm and ideal algorithm) together with the simulated locally greedy case. The figure shows that these abstract limiting cases provide a good hint for the actual system performance.

We can see that in both cases (a) and (b), total system utilization is significantly increased as γ goes from zero (no compensation) to 0.7. Further increase of γ does not increase the utilization,

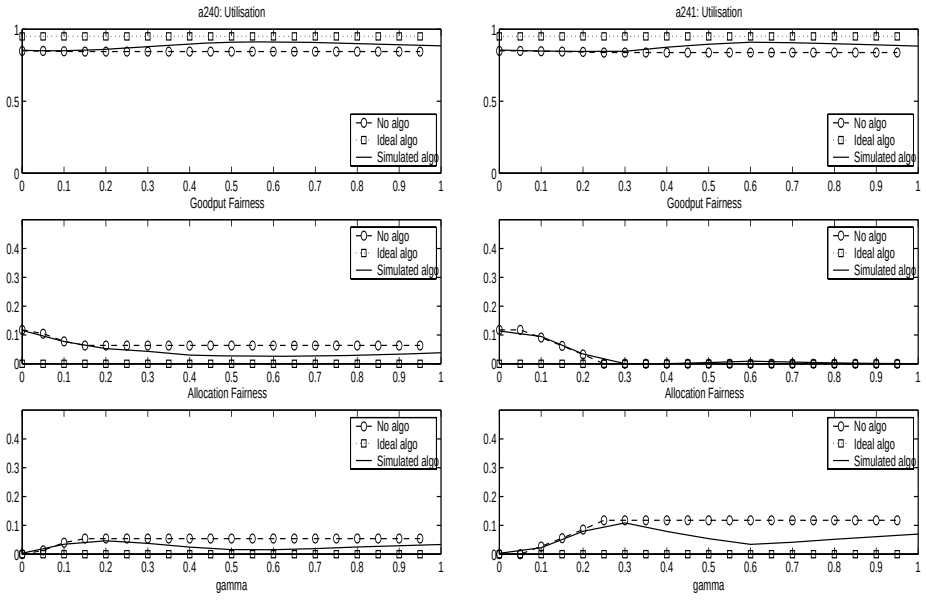


Figure 5.8: System performance, (a) $\beta_l = 0.5$ (b) $\beta_l = 1$, four users.

in fact, it can slightly decrease it. This is attributed to the fact that the specific user behaviour considered in this paper works better when the speed of compensation is not too fast (i.e., γ is lower than 1).

The fairness curves show that the algorithm, while keeping the increase in utilization, can significantly improve the goodput fairness of the system. It is seen that the fairness in goodput and allocation run counter to each other: improving one makes the other worse. For $\beta_l = 1$, almost complete fairness can be achieved for $\gamma > 0.3$. The figure clearly illustrates that it is possible to improve utilization and fairness simultaneously. This is possible because our architecture encourages efficient use of resources.

5.8 Summary

We have presented a scheme for the resource allocation of the capacity of a frame-based wireless base station with its performance evaluation. We have described a novel modular architecture where the fair master-scheduler does not explicitly take into consideration the channel state of individual users, and we have shown how a wireline fair scheduling algorithm can be simply extended to compensate for lost capacity after losses occur. We have extended the master-scheduler with compensation for unused capacity, so a user can optimize on its own when to relinquish service in the hope of more service later. We have given one possible way of user behaviour and shown by simulation that the pair of master scheduler and user decision rule can work together to improve both the total system utilization and the goodput of individual users.

In addition, we have introduced metrics by which the performance of the user algorithm can be assessed, and we have defined the utilization and fairness metrics by which the system can be analyzed. We have shown two simple abstract user behaviours, and we have implemented the locally greedy user algorithm in a packet-based simulator. We have shown that these abstract user behaviours can be handled analytically and provide a good approximation for the simulated locally greedy behaviour.

Chapter 6

A Novel Scheme to Interconnect Multiple Frequency Hopping Channels into an Ad Hoc Network

6.1 Introduction

So far, we have investigated packet-switched networks that all possessed a fixed infrastructure. Using wireless technologies, it is possible to design networks where such a pre-installed fixed infrastructure is not present. This type of networks are referred to as ad hoc networks. The MANET (Mobile Ad hoc Networking) working group of the IETF has been formed to study the protocol issues involved in such networks. The vision of the MANET working group [16] "is to support robust and efficient operation in mobile wireless networks by incorporating routing functionality into mobile nodes. Such networks are envisioned to have dynamic, sometimes rapidly-changing, random, multihop topologies which are likely composed of relatively bandwidth-constrained wireless links."

Frequency hopping spread spectrum radio technology [75] possesses a number of advantages that has motivated its selection in many radio systems. These advantages include robustness against interference, fading and noise, simplicity and low cost of implementation. A key advantage is that a number of such systems can be independently operated in the same coverage area with limited interference. There is no hard capacity limit for the number of interferers. Increasing their number results in a graceful degradation of performance.

Specifically, Bluetooth [10, 38] is one of the technologies that makes good use of the advantages of frequency hopping, as it has been designed to allow a large number of channels to co-exist in the same coverage area. Bluetooth is primarily intended as a cable replacement radio technology, using a short range (10m) radio interface designed to facilitate the development of very small and cheap implementations. Thanks to the frequency hopping radios, the system is indeed robust against interference caused by other Bluetooth and non-Bluetooth interferers in the same band [98].

When a large number of frequency hopping channels are present, the question of channel establishment and synchronization must be addressed. In the case of Bluetooth, devices have to synchronize using a paging procedure to establish the channel referred to as a piconet. The node initiating the procedure becomes the master of the piconet. The formation of the piconet takes a

relatively large overhead of several seconds, but makes data transmission straightforward once the piconet is established. This is in harmony with the requirements of cable replacement applications where a connection needs to be set up rarely, typically only once when the application is started or re-started. Once the piconet is established, the frequency hopping sequence is derived from the clock and address of the master node. The timing synchronization is defined by the transmissions of the master.

The channel establishment procedure makes it possible to set up multiple frequency hopping channels in the same coverage area. In Bluetooth, the hopping sequence is dependent on the master, which is why each piconet is using a different hopping channel. Although there can be a certain amount of interference, this provides a good separation of the radio channels. This also provides a logical separation since devices in different piconets do not even have to know about each other at all. Devices in the same piconet, on the other hand, need to be co-ordinated. This is performed by the master node using a centralized polling-based scheduling mechanism.

Once we have a large number of devices capable of communicating over a number of independent frequency hopping channels, it becomes a natural requirement to be able to connect them into a single network. This step, however, is problematic. In the case of Bluetooth, it is theoretically possible to form a network even though the system has been optimized for cable replacement scenarios. The specification allows a device to be a member in multiple piconets and several piconets can be connected into a so-called scatternet. Figure 6.1 shows an example of such a scatternet where two laptops L1 and L2 and projector Pj, together with other accessories, are connected into a network. Such scatternet networks are made possible by the specification, but a number of important issues remain unresolved, such as how to decide about piconet membership and master roles (i.e., connection setup), how to route packets, how to schedule the presence of a node in multiple piconets, and how to discover and manage neighbours. These problems have to be resolved in extension protocols to the core Bluetooth specification. Research (see for example [C8, C9, 81, 94, 52, 78]) and specification work [11] is ongoing to address these issues.

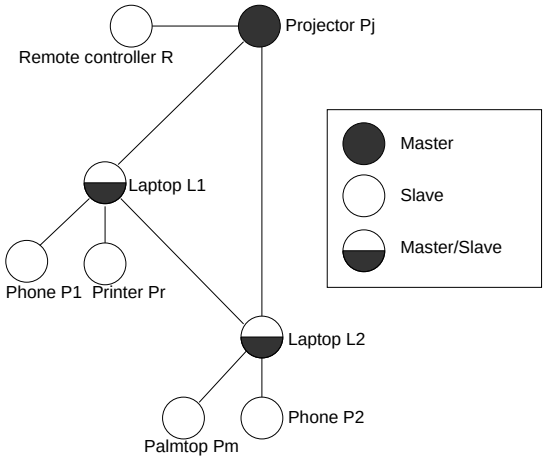


Figure 6.1: Example Bluetooth scatternet

In this chapter we suggest a new approach to interconnect multiple frequency hopping channels into an ad hoc network. We propose Multiple Frequency Hopping Channel communication (MFHC) to address this problem, and investigate the performance of MFHC ad hoc networks. Our approach avoids the use of a scatternet network, and allows nodes to communicate with all neighbours that are in radio range in a connection-less fashion. Our solution uses CSMA/CA

(Carrier Sense Multiple Access with Collision Avoidance) random access scheme [43] for each channel, with the extension that we allow a device to switch to a new frequency hopping channel (FHC for short) before each packet transmission. Each node has an associated home FHC that it follows by default. If a source node needs to send a packet to a destination node on the same home FHC, it uses the basic random access scheme on the common hopping channel. If, on the other hand, a source node needs to send a packet to a destination node that has a different home FHC than that of the source, then it switches to the home FHC of the destination and applies the random access scheme on the destination node's home FHC.

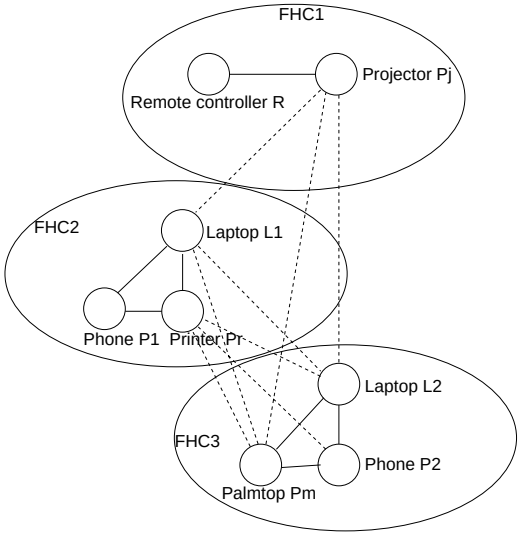


Figure 6.2: Example of the proposed MFHC scenario

Figure 6.2 shows the application scenario of Figure 6.1 employing the proposed MFHC scheme. The devices form three frequency hopping channels, denoted by FHC 1-3. Nodes within the same FHC can communicate with each other directly using CSMA/CA. This is shown by the solid lines. Nodes can also send data to another node in radio range in another FHC by switching to the destination node's home FHC. This is how communication between nodes connected by a dashed line (and between every other pair of nodes in radio proximity) can take place. As the figure suggests, the MFHC scheme avoids the complexity associated with establishing a scatternet and selecting master and slave roles, determining, optimizing and maintaining the topology and scheduling transmissions. MFHC also avoids multi-hop communication between neighbours. Instead, nodes can send packets to any of their neighbours by switching to the destination node's home FHC and following the well-known CSMA/CA scheme. The solution allows the formation of a connected ad hoc network, but at the same time keeps the advantages of using a single frequency hopping channel for a group of devices. To achieve this, MFHC requires a neighbour discovery and synchronization mechanism. One possible mechanism for neighbour discovery is described in [78].

The MFHC solution makes it possible to send data to a different node on another FHC, but it is clear that communication is more efficient when switching between FHCs is not needed. This raises the question of how to arrange the devices into groups using a common channel. One example is the one shown in the figure, but a number of other alternatives exist, including the important special case where each device has an associated FHC of its own. We will examine the performance trade-offs involved with grouping devices into FHCs. We will investigate static

FHCs, but we note here that it is possible to make the FHC selection algorithms dynamic, in which a node can change its home FHC membership based on traffic measurements.

The chapter is organized as follows. Section 6.2 reviews related frequency hopping systems and their networking capabilities. Section 6.3 describes the proposed MFHC solution. Section 6.4 presents three basic FHC configurations and compares them through a simple analytical model. Section 6.5 investigates MFHC via simulations. Section 6.6 concludes the chapter.

6.2 Related Work

A number of existing and proposed systems use frequency hopping spread spectrum radios, providing a limited networking capability. Here we provide a brief overview of such technologies in addition to Bluetooth that has already been introduced above. Table 6.1 summarizes the main features of the systems considered in this chapter.

Currently the most widely used ad hoc networking platforms are based on the IEEE 802.11 wireless LAN standard [43, 38]. At the MAC layer, multiplexing of traffic on a single channel is achieved by CSMA/CA. An RTS (request to send) - CTS (clear to send) - data - ACK four-way handshaking mechanism is defined. The RTS-CTS message exchange decreases the overhead of collision (when packets are long) and solves the hidden terminal problem [43]. IEEE 802.11 defines a number of physical layers, frequency hopping spread spectrum being one of them. However, communication is possible only in a single channel (between nodes in the same Basic Service Set in the 802.11 terminology). (Note that existing products that use the frequency hopping physical layer do not support fully distributed ad hoc operation even at a single channel, despite the fact that the standard allows this and defines a distributed time synchronization method. Instead, ad hoc operation is supported by products based on the direct sequence spread spectrum physical layer.) To use multiple channels, we have to have an infrastructure of connected access points. Without any infrastructure, it could be possible to use several independent hopping channels on the same coverage area to share the available spectrum, but only nodes on the same channel could communicate with each other. MFHC addresses this problem: it allows nodes in different channels to communicate.

The Hop-Reservation Multiple Access (HRMA) protocol is introduced in [87] for frequency hopping spread spectrum packet radios. The protocol uses a hop reservation and RTS-CTS handshake mechanism to guarantee collision-free operation even in the presence of hidden terminals. The protocol uses a designated frequency for control message exchange and requires timing synchronization over the whole network. By relying on this common channel that every node listens to, collision avoidance and hop reservation for data transmission can be achieved, so that multiple data transmissions use different frequencies. However, the requirement of synchronizing the whole network in time and using a single common signalling channel may imply performance and robustness bottlenecks.

The design concepts used in the High Frequency (HF) Intra Task Force (ITF) Communication Network are discussed in [25] employing frequency hopping spread spectrum radios. The proposal incorporates the Linked Cluster Algorithm that structures nodes into disjoint clusters making use of two TDMA frames that are synchronized over the whole network. Once the clusters are formed, a second procedure called Link Activation Algorithm controls how slots are allocated on the links. The available frequency band is divided into several sub-bands, and an independent network is formed in each sub-band. This makes it feasible to perform re-configuration of the network in one sub-band while communication can still continue in other sub-bands. However, the complexity and performance implications of re-configuration of the clusters and schedules are unclear.

The proposed MFHC scheme is novel in the way it establishes a connected ad hoc network when multiple unsynchronized frequency hopping channels exist on the same coverage area. The

System	Networking	Channel Setup	Resource allocation	Synchronization
Bluetooth Specification	Single piconet	Piconet formation	Centralized scheduling	Piconet-wide
Bluetooth Scatternet PAN	Connection-oriented multihop	Scatternet formation algorithm	Distributed scheduling	Piconet-wide
802.11/FH ad hoc	Single BSS	Distributed synchronization	On demand, CSMA/CA	BSS-wide
HRMA	Connection-less	Distributed synchronization	On demand, hop reservation	Network-wide
HF ITF	Connection-oriented multihop	Linked Cluster Algorithm	Scheduled (Link Activation Algorithm)	Cluster-wide, network-wide
MFHC	Connection-less	FHC selection algorithm	On demand, CSMA/CA	FHC-wide

Table 6.1: Summary of related work and MFHC with ad hoc frequency hopping systems

architecture combines many of the advantages of frequency hopping systems. It facilitates the use of low cost frequency hopping radios as in Bluetooth, it is based on a simple connectionless approach with on-demand resource allocation as in the case of IEEE 802.11, it enables networking between all devices as in HRMA and HF ITF, but without the need for a network-wide synchronization mechanism. MFHC can be adapted to frequency hopping physical layers with very different characteristics, e.g., to the physical layer of Bluetooth or 802.11 FH. As we make numerical investigations (Sections 6.4 and 6.5), we will use parameters that are typical of today's Bluetooth implementations, but is clear that MFHC is applicable with a number of other physical layer parameter settings as well.

6.3 Multiple Frequency Hopping Channel Communication (MFHC)

To interconnect multiple frequency hopping channels that co-exist on the same coverage area, we apply an adapted CSMA/CA scheme. Channel access within a FHC is based on the CSMA/CA approach used by the IEEE 802.11 protocol [43]. This means that a node that has a packet to send on the FHC first waits until the channel becomes free for at least a minimum period of time, which we refer to as GS (guard space). Communication may begin at fixed slot boundaries. To resolve collisions due to more than one stations sending at the same time, a contention mechanism is applied as follows. Each station has a contention window, CW, and chooses a random backoff value B from the interval $[0, CW - 1]$. In each slot when the channel is sensed free, the value of B is decreased if it is above zero. A node may transmit when the value of B reaches zero. If the transmission is successful, the value of CW is initialized to CW_{min}. If the transmission is unsuccessful, the value of CW is doubled unless it reaches CW_{max} and the transmission attempt will be repeated. This scheme ensures that collisions will be resolved after one or more stages of contention. (Note that in a practical implementation, it is important to limit the number of transmission attempts to avoid deadlock if the radio channel is down for any reason.)

We precede each packet transmission by an RTS-CTS message exchange, as in the 802.11 protocol. This handles the hidden terminal problem (the destination receives packets from a station that the source cannot receive from), and also decreases the overhead of contention in the case of long packets. In addition, the RTS-CTS message exchange provides a way for negotiation

of parameters for the subsequent data transmission.

This scheme can be extended for multiple FHCs, as shown in the example of Figure 6.3. Even though it is allowed for a node to switch from one FHC to another, we associate a default FHC with each node, which we refer to as the home FHC of the node. The figure shows two FHCs, where FHC 1 is the home of nodes A and B, FHC 2 is the home of nodes C, D and E. A node may temporarily leave its home FHC, as node B does to visit FHC 2 (B'), but it returns to its home FHC as soon as it has finished contention or transmission. To initiate a data transmission to a node, we need to switch to the destination node's home FHC and wait until the node is available and the channel is free.

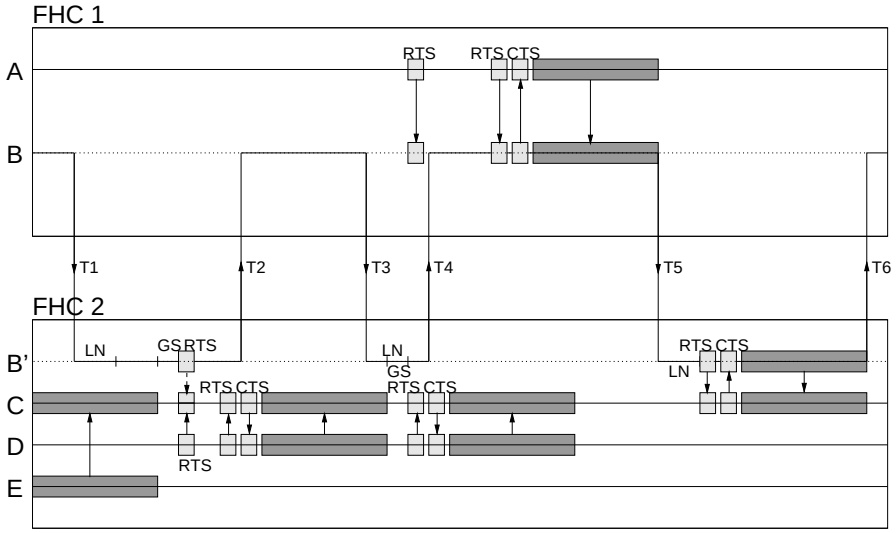


Figure 6.3: Example of Multiple Frequency Hopping Channel communication

When the destination node's FHC is different from the source node's home, then the source node has to switch between the source and destination FHCs during contention. This is illustrated in the figure, where node B wants to send a packet to node C in FHC 2. First, it switches to FHC 2 (becomes B' after transition T1) and listens on the channel for at least a fixed amount of time (denoted by LN (listen) in the figure). This is needed to synchronize to the channel and determine if there is an ongoing data transmission in the FHC or not. If there is an ongoing data transmission, as in the example, then B must wait until this transmission is over (and observe the guard space, GS) before sending an RTS. In the figure, node D also wants to send to node C, and after colliding with B at the first RTS transmission, it wins the contention in the second stage. B notices this when it hears the RTS from node D and waits until this data transmission is over. For this period of time, it switches back to its home FHC (transition T2). To determine when it can try again with a new RTS, node B uses its estimate of the length of the data transmission given in the RTS packet (this information is also given in CTS packets). Node B switches back to FHC 2 (transition T3) such that it spends the period of LN before its backoff counter reaches zero. In the figure, node D wins the contention once again, and B switches back to FHC 1 (transition T4). In the meantime, node A initiates a data transmission to node B which is unsuccessful because node B is away at that time. The RTS is retransmitted later, and the subsequent data transmission is started to node B. This delays node B switching to FHC 2 once again. However, when the transmission in FHC 1 is over, node B can immediately switch to FHC 2 (transition T5). After a period of LN has passed and FHC 2 is sensed free, node B sends its RTS which is

successfully received this time, allowing the consequent data packet transmission. Once this is over, node B switches back to its home FHC (transition T6).

The address and home FHC of a neighbouring node is known from a neighbour discovery mechanism. This is either based on a static configuration, or on beacon packets sent by the nodes [78]. Beacon packets can be sent at a dedicated frequency, or on a special frequency hopping sequence. In addition, beacon packets are sent on each FHC in order to synchronize the channel timing [43]. Note that while it is clear that we must ensure timing synchronization between two nodes that communicate with each other, MFHC does not require a network-wide synchronization mechanism. Here we do not consider the synchronization mechanism in detail, but we will consider the overhead of beacon packets used for channel synchronization in the analysis of Section 6.4.

Since MFHC makes it possible for the nodes to communicate with all neighbours within radio range in a connection-less fashion, it is possible to apply any of the MANET routing protocols [61] to extend connectivity over multiple hops. In this case, we use beacon packets to keep track of the neighbours. One issue that needs special attention in this case is that of broadcasts. Since MFHC uses multiple channels, a broadcast packet needs to be transmitted separately to neighbours on different channels.

6.4 Analysis of FHC Configurations

We now investigate the question of selecting the FHCs so that the performance of the communication is maximized. For this, we introduce three FHC configurations and compare their performance based on a simple analytical model. To enable the analysis, we first introduce a model for the contention mechanism. This will be followed by a system model that will be used for the subsequent performance comparison. In the comparison of this section we concentrate on the FHC configurations and simplify the details of the backoff mechanism, packet types and local retransmissions. Later in Section 6.5 we repeat and elaborate the analysis based on simulations of an implementation of the architecture.

6.4.1 Modelling of Contention

For our analysis we use a very simple performance model of the contention mechanism that captures the impact of the number of competing nodes on the time needed to resolve contention. In [8] and [9] the authors aim at modelling the behaviour of the IEEE 802.11 contention mechanism as closely as possible. Here we neglect most of the details and make some additional simplifying assumptions so that our performance model remains analytically tractable even in a multi-channel environment.

In our model, there is one round of contention in each slot. This means that we assume that the nodes sending the RTS packets get immediate feedback on the success or failure of the contention, and we neglect the possible loss and associated delay with the CTS packet. We also assume that each contending node is aware of an ongoing transmission on the channel and does not attempt to send an RTS during this period. Therefore in this model we consider only the slots when contention takes place. A node either sends an RTS in a slot, or defers sending its RTS, depending on whether its backoff counter has reached zero or not.

It has been observed [8, 96, 9] that the initial value of the contention window, CW_{min} may impact the overhead of the contention and its optimal value is dependent on the number of competing nodes. In our analysis we do not consider the question of setting the CW_{min} constant, instead we use the optimal setting of the contention window based on the number of contenders. (Note that this issue is considered in detail in [9] where an adaptation mechanism is proposed and it is shown that the performance of the adaptive setting is close to the optimal settings.)

Let the number of contending devices be denoted by k, and let the size of the contention window at each node be constant W. Using a simple Markovian model, it can be shown [9] that a single node transmits an RTS in a given slot with a probability of $\tau = 2/(W+1)$. In a given slot a new packet transmission is initiated when exactly one node transmits an RTS. Assuming independence between nodes, its probability is

$$P_{tx} = k(1-\tau)^{k-1}\tau(1-p). \tag{6.1}$$

where p is the probability of non-collision error (interference, fading, noise). Our purpose here is to find the value of τ (and W) that maximizes P_{tx}. Taking the derivative of the expression and solving it for zero, we get

$$\tau = \frac{1}{k} \tag{6.2}$$

and consequently $W = 2k - 1$. The probability of successful RTS transmission is then $P_{tx} = (1-1/k)^{k-1}(1-p)$. It is well known that the first factor in the formula goes to $1/e$ as k increases. We can thus approximate the probability as

$$P^*_{tx} = \frac{1}{e}(1-p) \tag{6.3}$$

In Figure 6.4 we show the probability of successful contention in a slot as a function of the number of competing nodes. The figure shows the results of simulation of the exponential backoff procedure with a fixed value of $CW_{min} = 8$, the value of P_{tx} for the optimal case with constant window as computed above, and the approximation P^*_{tx}. (In this case, $p = 0$ was used.) The simulated performance curve shows a slight increase which is due to the fixed initial contention window setting: when the number of contenders grows, the initial suboptimal setting no longer limits the performance. The optimal window performance P_{tx} gives an upper bound that tends to the simulated values of the backoff procedure as the number of nodes increase, similarly to the approximation P^*_{tx}.

In the following, we will use the approximation P^*_{tx} since it is close to the simulated backoff results especially as the number of nodes increases, and it gives an analytically tractable approximation which is independent of the implementation details and parameters of the contention procedure. (For simplicity, we will extend this approximation even for the $k = 1$ case.) With this approximation, the average number of slots it takes until one of the k nodes wins the contention is therefore

$$C = \frac{1}{P^*_{tx}} = \frac{e}{1-p}. \tag{6.4}$$

6.4.2 System Model

To model system performance, we introduce a network and traffic model, and compare a number of FHC configurations. Our primary performance metric will be the total system throughput. We will compare the throughput performance of three different FHC configurations.

To model a number of groups of devices using a common application over the same coverage area in an ad hoc networking scenario, we use a group-based traffic model: devices send most of their data to other members of the same group. The total of N nodes are divided into groups of size G. In our numerical analysis, we consider the extreme case where nodes within a group send packets to the members of the same group only. (Later in Section 6.5 we will investigate the effect of inter-group communication.) Sources are assumed to be greedy, which means that sources always have a packet to send. Before each packet transmission, the destination is chosen randomly and independently according to a uniform distribution from the other nodes in the same group. Each of the N nodes are within transmission range of each other, so transmissions

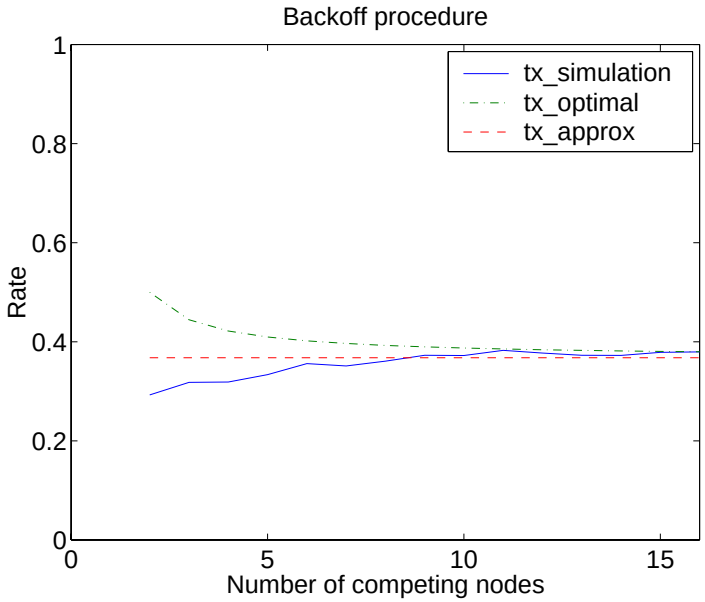


Figure 6.4: Simulated and analytical performance of contention

in different groups at the same time and same frequency collide.

We assume that transmission errors can be detected by an ARQ (Automatic Repeat reQuest) protocol but the details of this protocol are not considered here. In the analysis we assume that there exists a segmentation and reassembly mechanism, and the ARQ protocol retransmits the errored segments only. Therefore we model the additional load caused by retransmissions through the increase of the packet length by a factor of $1/(1-p)$, since $1-p$ is the success rate of data segment transmissions, where p is the transmission failure probability for a segment. (In the three configurations that we consider below, we will denote this probability by p_c, p_g and p_d.) We also assume segments of one slots in length, where a slot corresponds to the time the channel remains at one frequency hop (similarly to the Bluetooth system [10]). Furthermore we have an RTS packet that also takes one slot and we approximate the non-collision error probability of sending an RTS packet with that of sending a segment of one slot in length (that is, p). These assumptions are not essential to the MFHC proposal, and are used to facilitate the analysis in a potential application scenario.

We distinguish three different FHC configurations based on the set of nodes that have a common home FHC. In the *common* FHC case the same single channel is used by all of the N nodes. This will be our reference case where devices do not need to switch to a different FHC. In the *device* FHC case there is a separate FHC for contention and data transfer for each device. In this case, for each destination a node has to switch to a new FHC. The third FHC configuration that we investigate represents a compromise between the two extremes. In the *group* FHC case, every group of G nodes has its own FHC for contention and data transmission. Since in our traffic model of this section packets are sent only within the group, therefore nodes do not have to switch to a different FHC in this case, either.

Figures 6.5 - 6.7 illustrate the three cases. The dark rectangles represent the data packets sent on a given hopping channel, while the lightly shaded rectangles represent contention periods, with the arrows showing the direction of the data transmission and the contention. The winner

of the contention is marked by a solid arrow, while other contenders are marked with a dashed arrow. Note that a device may simultaneously compete to transmit to other nodes while receive RTS packets. This is achieved by switching between transmitting an RTS and receiving, as illustrated in the example of Figure 6.3.

In the following we will analyze each of these configurations separately.

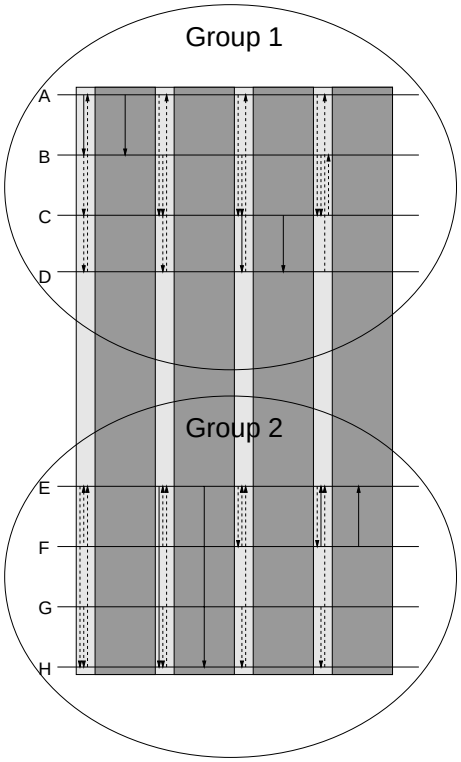


Figure 6.5: *Common* FHC: the same channel is used by all of the nodes.

6.4.3 Common FHC

In the common FHC case each device communicates on the same single frequency hopping channel (Figure 6.5). The channel is occupied by alternating transmission and contention periods.

To find an approximation for the system throughput, we have to consider the length of the data transmissions and the length of the contention periods. The length of the data transmissions is taken to be constant L_0. To find an approximation for the time spent with contention, we use the results of Section 6.4.1. The approximate average time until one of the nodes wins the contention is $C_c = e/(1-p_c)$ where p_c is the non-collision error probability when sending an RTS packet. In this analysis we only consider errors caused by interference, but in the *common* FHC case there is only one channel and therefore in this model we have $p_c = 0$.

From this, the load on the common frequency hopping channel (i.e., the fraction of time spent with packet transmission, including retransmissions) is

$$\Lambda_c = \frac{L_0}{L_0 + e}. \tag{6.5}$$

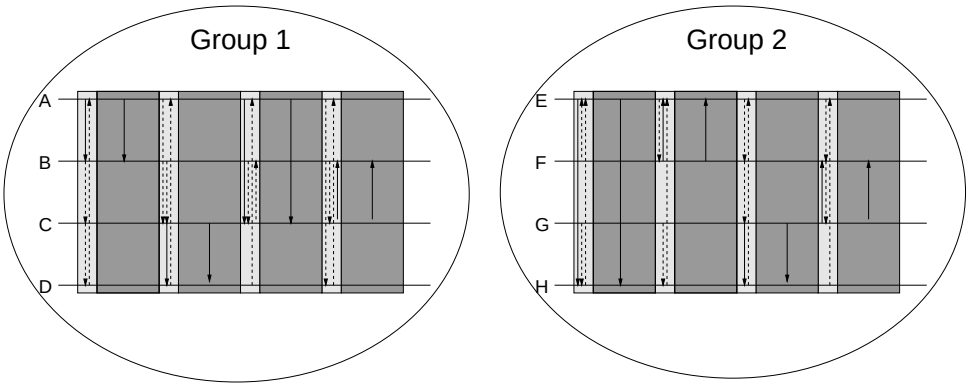


Figure 6.6: *Group* FHC: every group has its own channel.

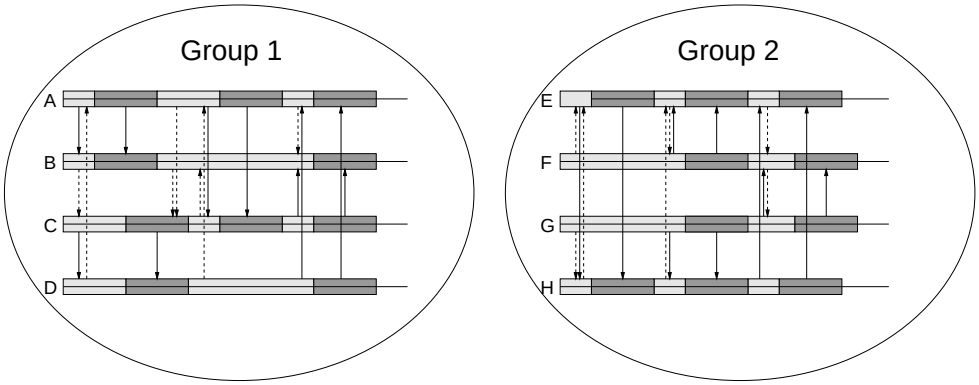


Figure 6.7: *Device* FHC: there is a separate channel for each of the nodes.

The traffic offered to the channel by a single node (i.e., the fraction of time spent with packet transmission, including retransmissions) is therefore:

$$\lambda_c = \frac{\Lambda_c}{N} = \frac{1}{N}\left(\frac{L_0}{L_0 + e}\right). \tag{6.6}$$

To find the total throughput, we also take into account that the channel synchronization must be maintained. This requires the exchange of packets that consume overhead. Here we consider that synchronization is maintained by the transmission of special single-slot beacon packets with a base period of T_b slots. This decreases the capacity of the channel by a factor of $1 - 1/T_b$. The total throughput is then

$$\Theta_c = \Lambda_c(1 - 1/T_b) \tag{6.7}$$

measured in the unit of the capacity of a single frequency hopping channel.

6.4.4 Group FHC

The group FHC case is characterized by each group of G devices using a common frequency hopping channel for both contention resolution and data transmission (Figure 6.6).

We use a similar approach to find an approximation for the system throughput as in the previous subsection. We have to characterize both the length of the data transmissions and the length of the contention periods. To characterize the length of the data transmissions, we assume that each packet transmission takes L slots, where the amount of data transmitted corresponds to a constant L_0 slots.

$L > L_0$ because there may be errors on the channel causing retransmissions, making the transmission of a complete packet longer. We only consider the errors caused by interference and use an independent and identically distributed error model with a segment error probability of p_g. The extra load caused by the retransmissions are approximated as $L = L_0/(1 - p_g)$ (as described in Section 6.4.2).

To find an approximation for the time spent with contention, we use $C_g = e/(1-p_g)$ where p_g is also the probability that an RTS is lost due to non-collision error (interference in our model). The load on a single FHC can then be computed as:

$$\Lambda_g = \frac{L}{L + C_g} = \frac{L_0}{L_0 + e}. \tag{6.8}$$

The traffic offered by a single node then becomes

$$\lambda_g = \frac{\Lambda_g}{G} = \frac{1}{G}\left(\frac{L_0}{L_0 + e}\right). \tag{6.9}$$

We now approximate the probability of interference error, p_g. A single frequency hopping channel is disturbed by $N/G - 1$ other similar channels. Each channel hops on K different carriers independently in a pseudo-random manner. Since the channels are not synchronized with each other, a transmission in a single slot in one channel may disturb two slots in a different channel if the carriers collide. If we neglect the interference caused by RTS and CTS packets, the probability that a transmission of one slot is successful despite the interference caused by another channel with a load of Λ_g is $1 - \Lambda_g \frac{2}{K}$. The error probability can then be approximated as

$$p_g = 1 - \left(1 - \Lambda_g \frac{2}{K}\right)^{N/G-1}. \tag{6.10}$$

The total throughput is obtained by summing the traffic in each channel, taking into account the synchronization overhead and that data transmission has an efficiency of $1-p_g$ due to errors:

$$\Theta_g = \frac{N}{G}\Lambda_g(1 - p_g)(1 - 1/T_b). \tag{6.11}$$

6.4.5 Device FHC

The device FHC is characterized by each node having a separate channel of its own (Figure 6.7). This means that each time a source node sends a data packet to any destination node, the source first has to switch to the FHC of the destination.

To find the traffic offered by a single node, we approximate the average time taken with data reception as follows. Similarly as above, a single packet reception takes $L = L_0/(1 - p_d)$ slots, where p_d is the segment error probability. A packet reception is preceded by a contention period. This would take on average $e/(1 - p_d)$ slots in general (since there is a non-collision error with probability p_d for an RTS).

However, contention is prolonged in this case for the following reasons. While waiting for incoming RTS packets initiating a data transfer, each node also has a packet to send at the same time. This means that a node has to switch between its own frequency hopping channel and that of the frequency hopping channel of its destination, as described in Section 6.3. The node participates in two contentions simultaneously, once as a potential transmitter and once as a potential receiver. Even if this could be done with 100% efficiency, this would double the time of the average contention. However, switching between the channels necessarily implies inefficiency. In addition, the contention window of source nodes are increased due to the fact that a destination node does not respond to an RTS when it has switched to a different FHC. The extent of these effects depends on many implementation dependent factors, such as the time needed to switch to a different FHC, and the setting of the maximum contention window. We approximate these effects by assuming that contention is prolonged by a factor of β due to the inefficiency incurred by switching between different channels. Our approximation for the contention period is therefore $C_d = \beta e/(1 - p_d)$. We have $\beta > 2$ since the length of contention is at least doubled. (We will investigate the value of β through simulations in Section 6.5.)

Due to symmetry between nodes and roles, each node spends the same amount of time with transmitting and receiving, and consequently transmits on average one packet for each packet reception. This follows that the fraction of time spent with reception at a given node is

$$\lambda_d = \frac{L}{2L + C_d} = \frac{L_0}{2L_0 + \beta e}. \tag{6.12}$$

Due to symmetry of the traffic model, λ_d is also the time spent with transmission by a given node.

Similarly to the previous subsection, the segment error probability can be found:

$$p_d = 1 - \left(1 - \lambda_d \frac{2}{K}\right)^{N-1}. \tag{6.13}$$

To find the total throughput, we have to take into account the synchronization overhead. Each node in a group has to synchronize to all other nodes in the group, giving a factor of $1 - G/T_b$. We can write the total system throughput as

$$\Theta_d = N\lambda_d(1 - p_d)(1 - G/T_b). \tag{6.14}$$

6.4.6 Performance Comparison

We now evaluate the performance of the FHC configurations based on the analytical model of the previous subsections. First, we plot the total throughput as a function of the total number of nodes N, see Figure 6.8. In the figure we use tentative parameter settings: the group size was fixed at 10 nodes, packet length was 12 slots, number of hop frequencies was set to 79, we used $\beta = 4$, and the synchronization overhead was not included. In the upper left box, we plot the offered traffic by a single node, that is, the fraction of time spent with data transmission by a node. First of all we can observe that this is constant for the *group* and *device* FHC configurations. To explain this, notice that the groups are logically independent and do not depend on each other except for the interference. The increase in the interference is shown in the upper right. Interference causes data transmissions to be longer, but it also prolongs the contention period by the same factor explaining why the offered traffic remains constant. (Note that in a given implementation the effect of packet losses may cause a different factor of increase for the data transmission time and for the contention. This may result in slight changes in the offered traffic as will be visible in the simulation results of the next section.) In the *common* FHC case, nodes share the same channel which causes the per node offered traffic to decrease as

the number of nodes increases.

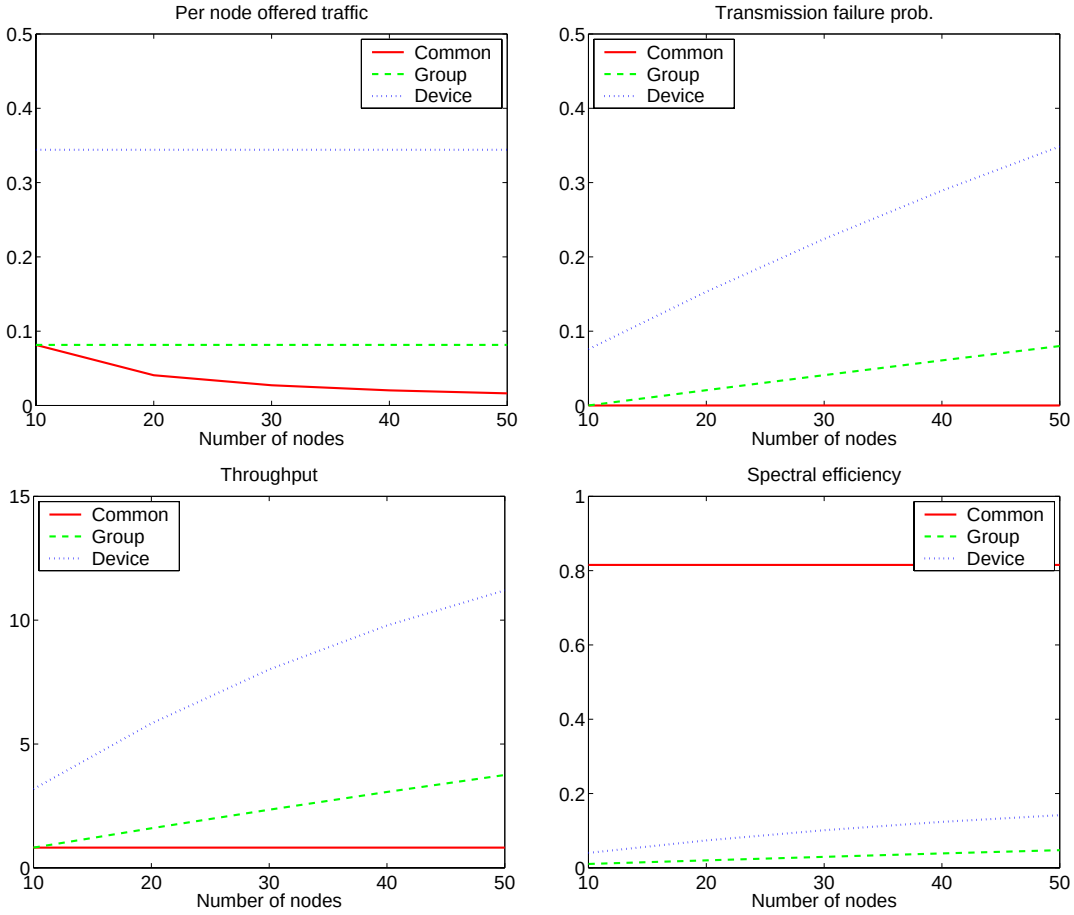


Figure 6.8: System performance, $G = 10, L_0 = 12, T_b = \infty$

The figure in the lower left box shows the total throughput of the system measured in the unit of the capacity of a single frequency hopping channel. This is constant for the *common* FHC case since the total capacity of a single channel is used, and it is not affected by interference. In the *device* and *group* cases, the total throughput increases with increasing number of nodes. This is because the number of FHCs are increased providing multiplexing gain. The slope of the curves decreases though because of the increased interference. The *device* FHC case allows for a greater number of parallel data transmissions to be multiplexed than the *group* FHC case which allows only a single transmission per group. This explains the significantly higher total throughput of the *device* FHC configuration.

Also plotted in the lower right box is a measure of the spectral efficiency. We obtained this measure by dividing the total throughput by the number of hop carriers, K. If all carriers were continuously transferring data, this measure would yield 1; its value therefore represents the efficiency of utilizing the available spectrum. We made an exception in the *common* FHC case, where we did not divide the total throughput by K, since only a single common channel is used in this case, which could - hypothetically - span even the whole available spectrum without causing any interference. In the rest of the cases, this is not possible since many channels need to be multiplexed that could interfere with each other.

The results show that the spectral efficiency is highest in the *common* FHC case, and it is lower for the other configurations. This observation can be interpreted as follows. If a single common high-speed channel can be used by all devices on an on-demand time-division basis, it can give a much more efficient usage of the available spectrum than dividing it into many uncoordinated low-speed channels. We have to keep this in mind when considering the other configurations employing multiple uncoordinated frequency hopping channels. However, a high-speed common channel may be difficult or costly to realize in practice. Note also that frequency hopping radios naturally lead to the use of multiple channels rather than a common channel of higher bandwidth. A full comparison of these two cases, involving other aspects such as hardware limitations, cost, radio propagation and error characteristics, is out of the scope of this chapter.

Figure 6.9 shows the dependence of the traffic offered by a node on the group size and packet length, with only a single group present. (Note that the offered traffic determines the total throughput.) We can see that the *device* FHC case offers a constant per node offered traffic, while the *group* and *common* FHC cases (which are identical in this scenario) yield a decreasing per node offered traffic. The reason for this is that the *device* FHC configuration allows multiplexing of data transmissions within a group. To compare the two curves at $G = 2$, notice that we have $\beta > 2$, which follows that $\lambda_d < \lambda_g$. This means that for a group of two nodes, the *device* FHC is necessarily less efficient than the *group* FHC. Depending on the implementation dependent value of β, the two curves must intersect each other, marking the group size where the *device* FHC configuration is equally efficient as the *group* FHC. By comparing the two graphs for long and short packet size, we can observe that the intersection point is also dependent on the packet size. When packets are shorter, the effect of backoff overhead is increased, therefore the per node offered traffic (and the total throughput) is lower.

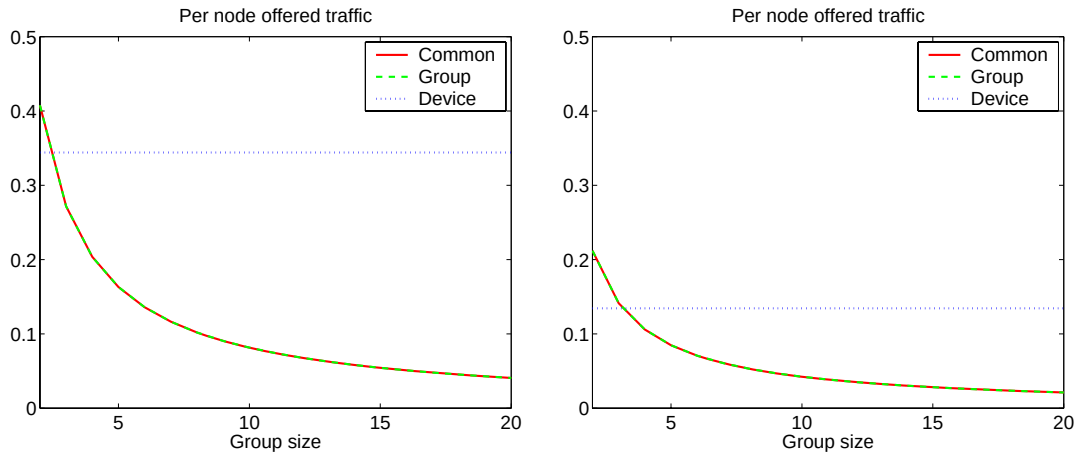


Figure 6.9: Dependence of the per node offered traffic on the group size and packet length. $N = G, T_b = \infty$. On the left $L_0 = 12$, on the right $L_0 = 2$.

Figure 6.10 shows the dependence of the total throughput on the synchronization overhead. This overhead depends on the accuracy of the clocks that are used: the less accurate they are, the more frequently we need to send beacon packets to keep the synchronization. The figures plot the base beacon sending period. The figures show that the *device* FHC case is the most sensitive to synchronization overhead, especially for higher group size. This is attributed to the fact that in this case, a node in a group has to synchronize to all other nodes in its group to be able to send data, while in the other cases nodes have to synchronize to one channel only. Note also that we took a very conservative computation for the synchronization overhead, since only a singe slot was wasted for a beacon packet. In a practical implementation, however, this

overhead might be much higher, which further emphasizes that the *device* FHC configuration is very sensitive to accurate synchronization.

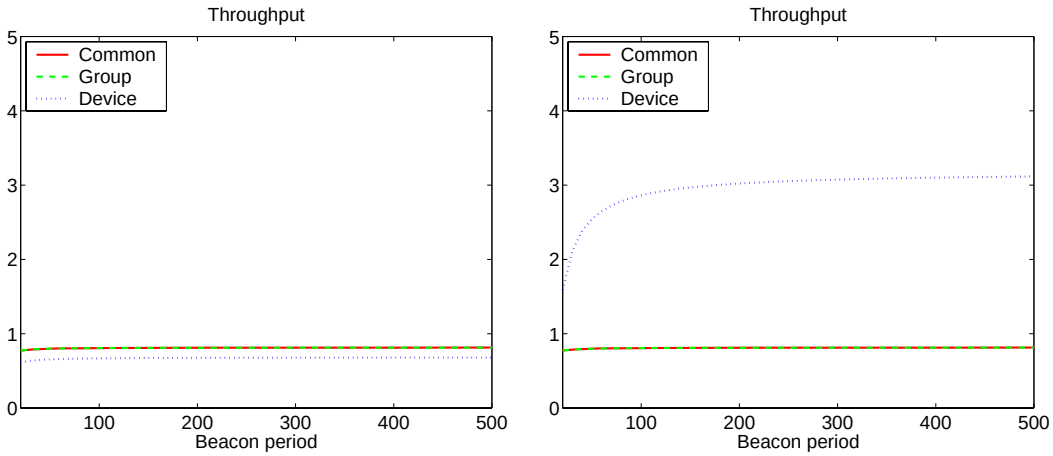


Figure 6.10: Dependence of system throughput on the beacon period and group size. $N = G, L_0 = 12$. On the left, $G = 2$, on the right, $G = 10$.

6.5 Simulation Study

To investigate the performance of the implementation of different FHC configurations, we have implemented the MFHC scheme in a packet level simulator [39]. Figure 6.11 shows the architecture of the simulator. The physical layer consists of a packet collision detector which determines the reception status of every individual packet. Each node has an associated FHC object in the physical layer (this association is shown by the dashed lines). The link layer representation of each node connects to exactly one FHC in the physical layer at a time, the one that it follows at the given moment as determined by the MFHC protocol implementation in the link layer (this connection is shown by the solid lines).

We consider scenarios where all nodes are within radio range of each other, which represents the worst case in terms of interference. In the physical layer model, packets can be lost due to interference (i.e., two or more packets are sent on the same frequency at the same time) or collision (i.e., an RTS packet collides with another packet sent to the same destination), otherwise they are delivered correctly. In the link layer, we model the contention mechanism as described in Section 6.3. FHCs are independent of each other using a pseudo-random frequency hopping pattern. We have implemented a segmentation and reassembly mechanism, and an ARQ protocol that gives feedback on the reception status after each segment. Lost segments are retransmitted immediately. Packets can be sent at the beginning of a slot. The slot timing is aligned to frequency hopping: there is a guard time at the end of each slot to allow devices to tune to a new frequency. Table 6.2 lists the parameters used in the simulations. Note that the channel capacity and the number of hop frequencies were selected to reflect the constraints of the 2.4GHz ISM band, and the other parameters were selected to reflect the current capabilities of typical Bluetooth implementations.

First we investigate the extent of multiplexing that can be achieved by using the *group* and *device* FHC configurations. Figure 6.12 shows the per node offered traffic and total system throughput as the number of groups, each with ten member nodes, are increased. The results are in accordance with the analysis of Section 6.4, showing that the *device* FHC configuration

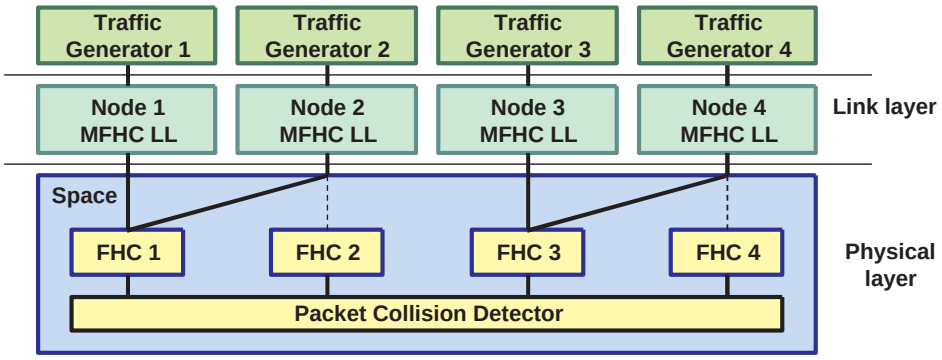


Figure 6.11: Simulator architecture

Channel capacity	1 Mbit/sec
Slot length	1 ms
Hop frequencies	79
Segment length	1, 2, 3, 4 or 5 slots
Segment header length	164 bits
Guard time for frequency hopping	0.1 ms per slot
Length of RTS, CTS, ACK packets	1 slot
Minimal contention window	8 slots
Maximal contention window	64 slots
Synchronization overhead	0 (not considered)
Listen time on new FHC	6 slots

Table 6.2: Simulation parameters

increases the total system throughput by a factor of two. However, the multiplexing gain that is achieved by the *device* FHC is only present with large group sizes. Figure 6.13 shows that with a group size of two nodes, the *device* FHC case actually performs worse by about 30%, because it is less efficient in contention and can not make use of multiplexing.

Figure 6.14 investigates the differences between small and large groups. The results follow the same trend as the analytical model shown in Figure 6.9. Fitting the formulas of Section 6.4 to the simulation results, we can approximate the value of β, which shows the inefficiency of contention in the *device* FHC case. We get a value of $\beta = 16$ in the case of 1500 byte packets and $\beta = 8$ in the case of 250 byte packets. These values are much greater than the minimal value of 2, showing that the switching of FHC during contention introduces a significant amount of extra overhead. Note also that this factor is not constant: in the case of long (1500 byte) packets and minimal group size of 2 nodes, the *device* FHC becomes more efficient (β decreases to 8 in this case). When there are only two nodes in the group, the intended destination node does not communicate with other nodes. That would cause failed RTS attempts whose number is much bigger in the case of long packets, which explains why we see this effect to a much greater extent in the case of long packets.

So far we have allowed traffic only within a group in our model. We now extend our traffic model to investigate the effect of traffic between the groups as well. In the extended model, with a probability p_{ng} a nodes chooses its destination from all the other nodes in the network, not just

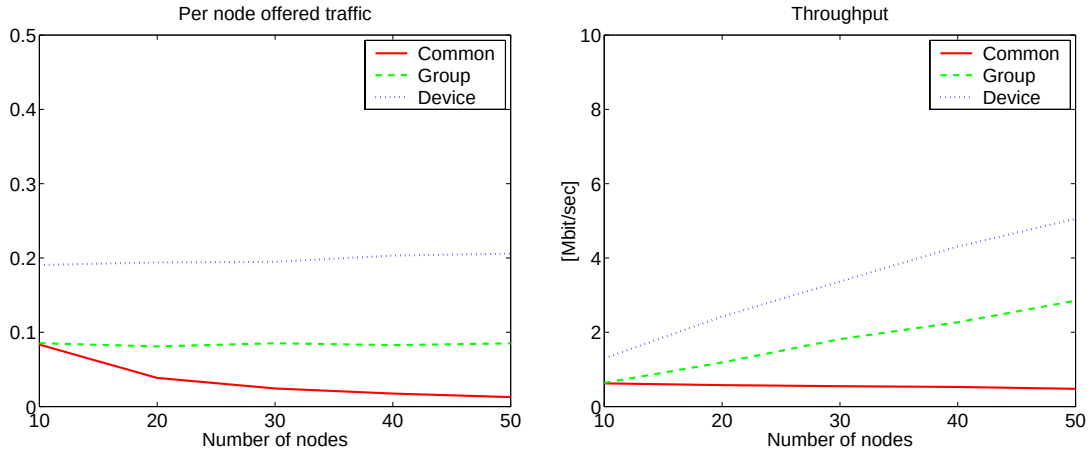


Figure 6.12: Per node offered traffic and total throughput as the number of groups are increased. Group size is fixed at 10.

its own group. Figure 6.15 shows the throughput performance as a function of the probability p_{ng} which determines the non-group traffic. The *device* FHC configuration is not sensitive to this change since it does not depend on the formation of groups. It only shows a slight throughput decrease in the case when the group size is two, which is explained by the reasoning above in the previous paragraph. The *group* FHC shows a decrease in both small and large group sizes, but the decrease is much more significant when the group size is small. The reason for this is that when the group size is high, there is a higher chance that at least one of the potential destinations is available, and so the channel can be utilized. In both cases, the results show that the *device* FHC configuration gives higher performance in the case of heterogeneous traffic, that is, when there is significant traffic between the groups.

Finally we observe the effect of changing the traffic pattern within a group to model a client-server application (with no traffic between the groups). In this case we designate one node in all groups to be a server and the other nodes in the group to be clients. All nodes remain greedy as before in that they always have a data to send, but with a constant probability p_s, the clients choose the server as their destination. Figure 6.16 shows the total throughput as the constant p_s is increased from 0 to 1 (server-client traffic only). In this experiment the total throughput of the *group* FHC configuration remains constant since this is determined by the capacity of the group channel. On the other hand the performance of the *device* FHC configuration decreases to that below the *group* case. When there is only server-client traffic, the *device* FHC case can not achieve multiplexing gain, and it uses a less efficient contention scheme than the *group* FHC which explains its lower performance.

6.6 Summary

We have proposed Multiple Frequency Hopping Channel communication (MFHC), a scheme that forms a connected ad hoc network from multiple frequency hopping channels. Our scheme relies on the notion of home FHC. Each device participating in an ad hoc network has a home FHC which determines the frequency hopping scheme it follows whenever it is not transmitting at another FHC. To transmit to a particular device, it is necessary to switch to that particular device's home FHC, listen to the channel and resolve contention. The difference from a traditional random access scheme is that besides the possibilities of success or collision, a third option is

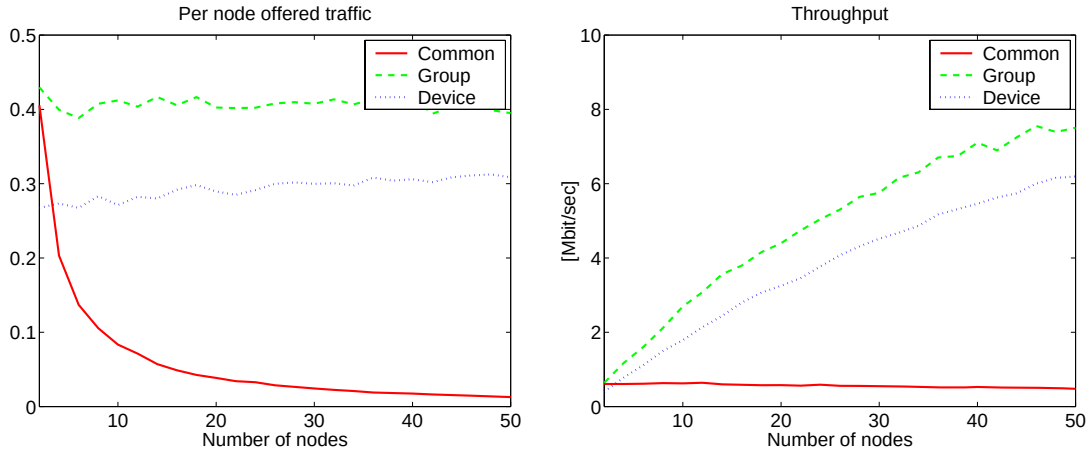


Figure 6.13: Per node offered traffic and total throughput as the number of groups are increased. Group size is fixed at 2.

that the destination is "away" at another FHC.

Besides allowing ad hoc networks to benefit from the advantages of frequency hopping, this scheme increases their throughput compared to using a single channel only, but it requires additional coordination. We have investigated the impact of this additional coordination on the system's performance using analytical and simulation tools. In particular, we have compared the extreme case of MFHC, where each device has its own distinct FHC, to a reference case where the entire ad hoc network uses the same single FHC. The results show that the former case (*device* FHC) provides significantly higher total throughput than the reference case (*common* FHC).

We have also analyzed a case where subsets of an ad hoc network form a partially closed communication group in the sense that members of one group communicate mostly with other members of the same group and rarely with other nodes of the ad hoc network. This scenario may be typical in some realistic ad hoc networks. We have shown how MFHC can adapt to this case such that members of one group share the same home FHC. This case, referred to as *group* FHC, represents a compromise between the *device* FHC and *common* FHC cases. We have shown that it is especially well suited to server-client type traffic patterns, but it is ill-suited for heterogeneous traffic patterns. The *group* FHC configuration makes the contention mechanism more efficient and it requires less overhead for channel synchronization, in exchange for lower multiplexing gain and consequently lower total throughput.

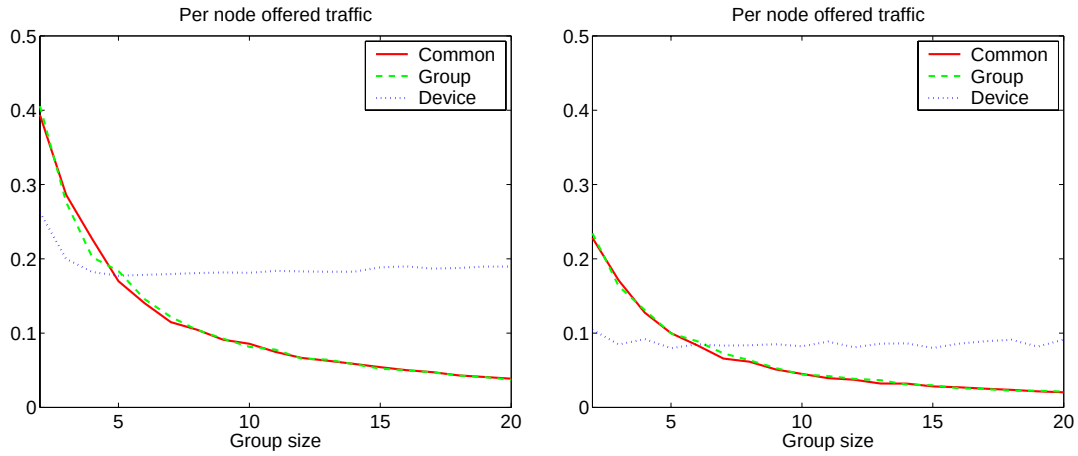


Figure 6.14: Per node offered traffic in the case of a single group. On the left packet length is 1500 byte, on the right packet length is 250 byte.

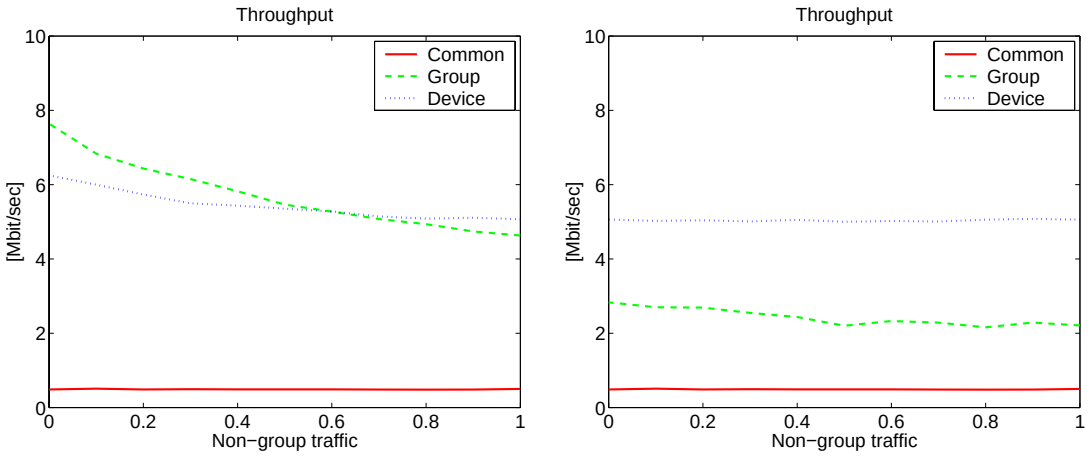


Figure 6.15: The effect of non-group traffic on the total throughput. On the left group size is 2, on the right group size is 10. The number of nodes is fixed at 50.

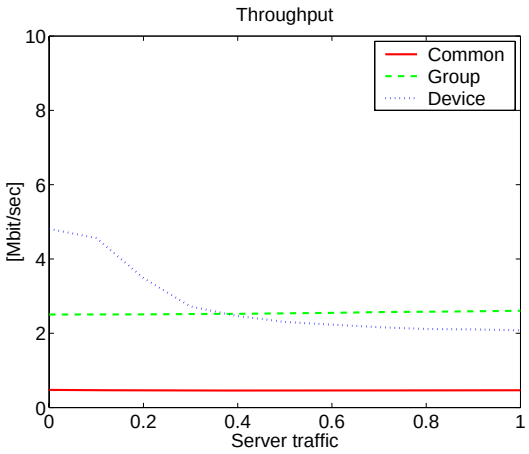


Figure 6.16: The effect of server-client traffic on the total throughput. Group size is fixed at 10, number of nodes is 50.

Chapter 7

Conclusions

7.1 Contribution of Thesis

In the dissertation we have addressed some of the challanges of supporting data traffic in communication networks, mobile networks in particular. The unpredictable nature of the user traffic demand and the erroneous nature of wireless air interfaces motivated the design of a number of new control mechanisms.

We have examined one motivation, the characterisation of dynamic traffic demand, in details. We have applied the concept of peakedness for traffic characterization. We have given the formulation of peakedness in discrete time, and we have also given a number of practical considerations on its application. We have applied the peakedness measure for a number of measurement traces, showing a great deal of variability depending on the application. We have also given a fitting technique that fits a Markovian model to a peakedness curve.

After characterizing the variability of traffic, we have investigated control mechanisms that work adaptively depending on the traffic and channel characteristics. We have investigated a single wireless link and proposed a new automatic repeat request mechanism, and we have also investigated the resource allocation of a wireless base station with many terminals.

We have proposed a new type of selective repeat retransmission (ARQ) protocol in the HIPERLAN/2 wireless LAN architecture. In this case, dynamic adaptation can select the appropriate negative acknowledgements and determine the amount of ARQ signalling capacity. This adaptation is based on the co-operation of the terminals and the base station scheduler. We have presented simulation results that show that the proposed scheme with dynamic adaptation is more efficient than static schemes in terms of thoughput and delay.

In the case of the resource allocation of a wireless base station, we have analyzed the interaction between a number of control mechanisms. The retransmission protocol works on each individual wireless link at the link layer. The transport layer (TCP) provides another level of retransmissions, and based on the observed capacity, it adapts the sending rate of the end source. The resource allocation mechanism employs another level of adaptation that compensates for the losses that occur on the air interface. Our aim with the compensation mechanism was to control the trade-off between system utilization and fairness. This trade-off arises because — unlike in fixed networks — the amount of service given to a flow is not the same as the amount of service that it gets due to the location-dependent channel errors.

We have made a number of observations in our simulation-based analysis. We have found that the link-layer ARQ mechanism can work independently from the transport-layer TCP retransmissions since the link-layer in our case is much faster. We have investigated the effect of persistency of the link-layer ARQ mechanisms. We have found reliable (fully persistent) retransmissions at the link layer improve efficiency and make the fairness of the resource allocation

controllable. On the other hand, if the link-layer retransmissions are given up after a threshold number of trials, the effect of packet losses can be that the TCP congestion control algorithms are triggered, which in turn can completely change the fair resource sharing. We have also shown that a simple compensation mechanism can significantly improve fairness at the cost of slight decrease in the system utilization. The reduction of utilization is slight in the case of independent channel losses, and increases with the burstiness of the loss model.

We have extended this scheme with two additional levels of adaptive control. We have extended the scheduler in the base station with a compensation mechanism for unused capacity. The base station gives additional capacity to a user (terminal) later if it requests less capacity than the amount that it could receive. This encourages users to optimize their channel usage on their own. Another adaptive mechanism at the user makes measurements and estimations of the channel, and when it predicts that the channel will be erroneous, it relinquishes its capacity request. The working of the user is determined by the channel characteristics: errors are typically bursty in nature, which can be utilized in the resource requesting mechanism. We have extended the scheme with an optional third adaptive mechanism that helps users set the sensitivity of their algorithm based on a one-bit feedback from the base station. We have used simulations and analysis to evaluate this architecture. We have shown that this scheme encourages the efficient use of the capacity. For this reason, the system utilization and fairness can be improved together. We have also introduced two abstract user behaviour for the analysis and used them to approximate the performance of the system.

Finally, we have addressed the question of a mobile ad hoc network using frequency hopping spread spectrum radios. In these networks, all nodes may be mobile, and communication may take place using multiple hops. Due to the frequency hopping radios however, two devices need to use the same frequency hopping channel in order to communicate. This requires some sort of co-ordination mechanism. We have proposed MFHC (Multiple Frequency Hopping Channel Communication) to enable devices communicate even when many channels are used simultaneously in the network. Devices switch between the different channels and communicate using the CSMA/CA principle.

We have used simulation and analytical methods to compare the performance of different frequency hopping channel configurations. The results have validated the feasibility of this type of novel ad hoc networking architecture. In particular, we have shown that the extreme case where each device has an associated frequency hopping channel can achieve high throughput performance due to its multiplexing gain. We have also shown that other configurations where a group of devices uses the same channel can also achieve high performance, and in some cases this is the most efficient scenario. Our results give guidelines on the performance differences of the different configurations.

7.2 Areas for Future Research

There are many ways to continue the research in the dissertation. Below we suggest some interesting areas of continuation.

Traffic characterization remains an open area for research. One interesting topic is the investigation of non-stationarity in the measurements. Another open area remains to develop suitable characterization methods that take into account the transport layer (TCP) feedback mechanisms that influence traffic behaviour.

Regarding dynamic retransmission protocols, in the future it is essential to carry on the analysis using measured channel characteristics. Such measurements could open the way for optimization of the adaptation mechanisms of the ARQ protocol.

In the resource allocation of a wireless base station, future research needs to verify the findings in a real implementation. The effect of implementation constraints need to be addressed. The

user behaviour algorithms have to be evaluated under a number of different channel conditions. Based on the results, a possible set of new use behaviour algorithms can be developed.

Another interesting area of future research is to provide delay bound based on certain assumptions on the wireless channel. Other scheduling principles can also be studied, and their performance characteristics can be investigated by further simulation as well as analytical methods.

Regarding ad hoc communication over multiple frequency hopping channels, also a number of new research directions remain. The dissertation has indicated a number of different channel configurations. In the future, protocols can be developed that are based on these performance findings and control the frequency hopping channel configurations. The protocols will have to be verified by further performance analysis. In addition, neighbour discovery and synchronization mechanisms need to be studied in more detail. The multihop networking scenario also requires further study.

The dissertation has addressed some of the design issues that arise as a consequence of the shift towards mobile data communication networks. We can be sure that in the years to come, this trend will continue to pose a large number of research questions.

Appendix A

Derivation of Peakedness Results

A.1 Peakedness in Discrete Time

We begin the analysis of peakedness in discrete time by refreshing the notation of Chapter 2. Time is discretized, and the amount of work arriving at epoch i is denoted by $w[i]$. We suppose and infinitely long past: $i = \ldots -1, 0, 1, \ldots$. This process is assumed to be stationary. The number $w[i]$ corresponds to the number of arrivals in epoch i, but it can be interpreted in a more general sense, as the amount of work arriving in epoch t. (The formulas below are valid even if $w[i]$ is not integer.) The first and second moments of $w[i]$ are $m_1 = \mathrm{E}\{(w[i])\}$, $m_2 = \mathrm{E}\{(w^2[i])\}$, which are independent of i due to the stationarity assumption. The autocovariance function of $w[i]$ is $k[s] = \mathrm{Cov}\{(w[i], w[i+s])\} = k[-s]$. It is easily seen that $k[0] = m_2 - m_1^2$.

The arrivals are offered to an infinite group of servers, where the service time T is a random variable. Its distribution is $t[1], t[2], \ldots$ on $1, 2, \ldots$. (It cannot take on zero value.) The complentary holding time distribution function of the service time is $F^c[x] = \sum_{s=x+1}^{\infty} t[s] = P(T > x)$ if $x \geq 0$ and $F^c[x] = 0$ if $x < 0$. Denote the service intensity by $\mu = 1/\mathrm{E}\{T\}$. It can be shown that $1/\mu = \mathrm{E}\{T\} = \sum_{-\infty}^{\infty} F^c[s]$. Autocorrelation function of F^c is $\rho_{F^c}[x] = \sum_{s=-\infty}^{\infty} F^c[s]F^c[s+x]$. It is seen that $\rho_{F^c}[0] = \sum_{s=-\infty}^{\infty}(F^c)^2[s]$.

A.2 Peakedness Computation

In each epoch there are $w[i]$ arrivals, and a server is assigned to each of them, where the service time of each server is independent and has a complementary holding time distribution of $F^c[x]$. The peakedness of the arrival stream corresponding to this holding time distribution is $Z\{F^c\} = \frac{\mathrm{Var}\{L[t]\}}{\mathrm{E}\{L[t]\}}$ where $L[t]$ is the number of busy servers at epoch t. The peakedness is independent of t due to the stationarity of the process. Our purpose here is to determine the peakedness by computing $\mathrm{Var}\{L[t]\}$ and $\mathrm{E}\{L[t]\}$.

Let us define the indicator

$$h(t, s, x) = \begin{cases} 1 & \text{if } s \leq t < s + x \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

The number of busy servers due to the arrivals at epoch s is

$$L_{s,w[s]}[t] = \sum_{i=1}^{w[s]} h(t, s, T_i[s]) \tag{A.2}$$

where $T_i[s]$ is the service time random variable assigned to the ith arrival of the batch at epoch s independently for each arrival; s and $w[s]$ are fixed. For each i, $P(h(t,s,T_i[s])=1) = F^c[t-s]$ due to the definition of $F^c[t-s]$. It follows that the distribution of $L_{s,w[s]}[t]$ is binomial, and

$$\begin{aligned} \mathrm{E}\left\{L_{s,w[s]}[t]\right\} &= w[s]F^c[t-s] && \text{(A.3)} \\ \mathrm{Var}\left\{L_{s,w[s]}[t]\right\} &= w[s]F^c[t-s](1-F^c[t-s]) && \text{(A.4)} \end{aligned}$$

For a fixed arrival stream $S : \ldots, w[-1], w[0], w[1], \ldots$, we can sum up for all s to get, using the independence of the service of the different arrivals,

$$\begin{aligned} \mathrm{E}\left\{L[t]|S\right\} &= \sum_{s=-\infty}^{\infty} w[s]F^c[t-s] && \text{(A.5)} \\ \mathrm{Var}\left\{L[t]|S\right\} &= \sum_{s=-\infty}^{\infty} w[s]F^c[t-s](1-F^c[t-s]). && \text{(A.6)} \end{aligned}$$

From this, we have

$$\begin{aligned} \mathrm{E}\left\{L[t]\right\} &= \mathrm{E}\left\{\mathrm{E}\left\{L[t]|S\right\}\right\} = \\ &= \mathrm{E}\left\{\sum_{s=-\infty}^{\infty} w[s]F^c[t-s]\right\} \\ &= \sum_{s=-\infty}^{\infty} \mathrm{E}\left\{w[s]F^c[t-s]\right\} \\ &= \sum_{s=-\infty}^{\infty} \mathrm{E}\left\{w[s]\right\}\mathrm{E}\left\{F^c[t-s]\right\} \\ &= m_1 \sum_{s=-\infty}^{\infty} F^c[t-s] \\ &= m_1 \mathrm{E}\left\{T\right\} \\ &= m_1/\mu. && \text{(A.7)} \end{aligned}$$

To compute $\mathrm{Var}\left\{L[t]\right\}$, we will need $\mathrm{E}\left\{\mathrm{Var}\left\{L[t]|S\right\}\right\}$ and $\mathrm{Var}\left\{\mathrm{E}\left\{L[t]|S\right\}\right\}$. Using eq. (A.6) and eq. (A.5),

$$\begin{aligned} \mathrm{E}\left\{\mathrm{Var}\left\{L[t]|S\right\}\right\} &= \mathrm{E}\left\{\sum_{s=-\infty}^{\infty} w[s]F^c[t-s]\right\} - \mathrm{E}\left\{\sum_{s=-\infty}^{\infty} w[s](F^c[t-s])^2\right\} \\ &= m_1\left(\sum_{s=-\infty}^{\infty} F^c[s] - \sum_{s=-\infty}^{\infty} (F^c[s])^2\right) \\ &= m_1(\mathrm{E}\left\{T\right\} - \rho_{F^c}[0]) \\ &= m_1(1/\mu - \rho_{F^c}[0]). && \text{(A.8)} \end{aligned}$$

$$\mathrm{Var}\left\{\mathrm{E}\left\{L[t]|S\right\}\right\} = \mathrm{Var}\left\{\sum_{s=-\infty}^{\infty} w[s]F^c[t-s]\right\}$$

$$
\begin{aligned}
&= \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} \mathrm{Cov}\left\{(w[s]F^c[t-s], w[r]F^c[t-r])\right\} \\
&= \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} F^c[t-s]F^c[t-r]\mathrm{Cov}\left\{(w[s], w[r])\right\} \\
&= \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} F^c[s]F^c[r]k[s-r] \\
&= \sum_{s=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} F^c[s+r]F^c[r]k[s] \\
&= \sum_{s=-\infty}^{\infty} \rho_{F^c}[s]k[s]. \qquad \text{(A.9)}
\end{aligned}
$$

Taking the two results eq. (A.8) and eq. (A.9) together, and using the fact that $\mathrm{Var}\{X\} = \mathrm{E}\{\mathrm{Var}\{X|Y\}\} + \mathrm{Var}\{\mathrm{E}\{X|Y\}\}$,

$$
\begin{aligned}
\mathrm{Var}\{L[t]\} &= \mathrm{E}\{\mathrm{Var}\{L[t]|S\}\} + \mathrm{Var}\{\mathrm{E}\{L[t]|S\}\} \\
&= m_1(\mathrm{E}\{T\} - \rho_{F^c}[0]) + \sum_{s=-\infty}^{\infty} \rho_{F^c}[s]k[s] \\
&= m_1/\mu + \sum_{s=-\infty}^{\infty} \rho_{F^c}[s](k[s] - m_1\delta[s]) \qquad \text{(A.10)}
\end{aligned}
$$

where $\delta[s]$ is the discrete time delta function ($\delta[s] = 1$ if $s = 0$, $\delta[s] = 0$ otherwise).

The peakedness of the arrival stream can now be expressed by eq. (A.7) and eq. (A.10) as

$$
\begin{aligned}
z\{F^c\} &= \frac{\mathrm{Var}\{L[t]\}}{\mathrm{E}\{L[t]\}} \\
&= 1 + \frac{\mu}{m_1} \sum_{s=-\infty}^{\infty} \rho_{F^c}[s](k[s] - m_1\delta[s]). \qquad \text{(A.11)}
\end{aligned}
$$

A.3 Single Holding Time for a Batch

The computation of peakedness is similar when we assign only one holding time variable to a batch, and assign one server to each arrival according to this common holding time.

In this case, eq. (A.2) becomes

$$
L_{s,w[s]}[t] = w[s]h(t, s, T) \qquad \text{(A.12)}
$$

as there is only a single random holding time. The value of $h(t, s, T)$ is 1 with probability $F^c[t-s]$ and 0 otherwise, so the expected value of $L_{s,w[s]}[t]$ is not changed but the variance is as compared to eq. (A.3), eq. (A.4):

$$
\begin{aligned}
\mathrm{E}\left\{L_{s,w[s]}[t]\right\} &= w[s]F^c[t-s] \\
\mathrm{Var}\left\{L_{s,w[s]}[t]\right\} &= (w[s])^2F^c[t-s](1-F^c[t-s]) \qquad \text{(A.13)}
\end{aligned}
$$

From this, we get the same result for $\mathrm{E}\{L[t]\}$, $\mathrm{Var}\{\mathrm{E}\{L[t]|S\}\}$ as eq. (A.7) and eq. (A.9).

But for $\mathrm{E}\{\mathrm{Var}\{L[t]|S\}\}$, we get a different constant:

$$\begin{aligned}
\mathrm{E}\{\mathrm{Var}\{L[t]|S\}\} &= \mathrm{E}\left\{\sum_{s=-\infty}^{\infty}(w[s])^2F^c[t-s](1-F^c[t-s])\right\} \\
&= \sum_{s=-\infty}^{\infty}\mathrm{E}\left\{(w[s])^2F^c[t-s](1-F^c[t-s])\right\} \\
&= \sum_{s=-\infty}^{\infty}\mathrm{E}\left\{(w[s])^2\right\}F^c[t-s](1-F^c[t-s]) \\
&= m_2(\mathrm{E}\{T\}-\rho_{F^c[0]}) \\
&= m_2(1/\mu-\rho_{F^c[0]}) \qquad \text{(A.14)}
\end{aligned}$$

This gives, using eq. (A.14) and eq. (A.9),

$$\begin{aligned}
\mathrm{Var}\{L[t]\} &= \mathrm{E}\{\mathrm{Var}\{L[t]|S\}\}+\mathrm{Var}\{\mathrm{E}\{L[t]|S\}\} \\
&= m_2(1/\mu-\rho_{F^c[0]})+\sum_{s=-\infty}^{\infty}\rho_{F^c}[s]k[s] \\
&= m_2/\mu+\sum_{s=-\infty}^{\infty}\rho_{F^c}[s](k[s]-m_2\delta[s]) \qquad \text{(A.15)}
\end{aligned}$$

and therefore the peakedness for single holding time for a batch, denoted by $\tilde{z}\{F^c\}$, is expressed using eq. (A.15) and eq. (A.7):

$$\begin{aligned}
\tilde{z}\{F^c\} &= \frac{\mathrm{Var}\{L[t]\}}{\mathrm{E}\{L[t]\}} \\
&= \frac{m_2/\mu+\sum_{s=-\infty}^{\infty}\rho_{F^c}[s](k[s]-m_2\delta[s])}{m_1/\mu} \\
&= \frac{m_2}{m_1}+\frac{\mu}{m_1}\sum_{s=-\infty}^{\infty}\rho_{F^c}[s](k[s]-m_2\delta[s]) \qquad \text{(A.16)}
\end{aligned}$$

We now determine the relation between $z\{F^c\}$ and $\tilde{z}\{F^c\}$ From eq. (A.11) and eq. (A.16) we can easily express $\sum_{s=-\infty}^{\infty}\rho_{F^c}[s]k[s]$ to get an expression connecting $z\{F^c\}$ and $\tilde{z}\{F^c\}$:

$$\frac{z\{F^c\}-1}{\mu/m_1}+\rho_{F^c}[0]m_1=\frac{\tilde{z}\{F^c\}-m_2/m_1}{\mu/m_1}+\rho_{F^c}[0]m_2 \qquad \text{(A.17)}$$

which yields

$$\tilde{z}\{F^c\}-z\{F^c\}=\left(\frac{m_2}{m_1}-1\right)(1-\rho_{F^c}[0]\mu). \qquad \text{(A.18)}$$

The first factor is zero if and only if the arrival stream has no batches, the second factor is zero only if and only if the holding time distribution is deterministic.

A.4 Geometrically Distributed Holding Times

If the random variable T is geometrically distributed, that is, $t[i] = \mu(1-\mu)^{i-1}$ $(0 < \mu < 1)$, and so $\mathrm{E}\{T\} = 1/\mu$, then we have

$$F^c[x] = \sum_{k=x+1}^{\infty} \mu(1-\mu)^{k-1} = (1-\mu)^x. \tag{A.19}$$

The autocorrelation function for $i \geq 0$ is expressed as

$$\begin{aligned}
\rho_{F^c}[i] &= \sum_{s=0}^{\infty} F^c[s]F^c[s+i] \\
&= \sum_{s=0}^{\infty} (1-\mu)^s(1-\mu)^{s+i} \\
&= (1-\mu)^i \sum_{s=0}^{\infty} (1-\mu)^{2s} \\
&= (1-\mu)^i \frac{1}{1-(1-\mu)^2} \\
&= \frac{(1-\mu)^i}{\mu(2-\mu)}
\end{aligned} \tag{A.20}$$

which gives

$$\rho_{F^c}[i] = \frac{(1-\mu)^{|i|}}{\mu(2-\mu)}. \tag{A.21}$$

Using formula eq. (A.11) we get the peakedness value for geometrically distributed holding times:

$$\begin{aligned}
z_{\mathrm{geo}}(\mu) &= 1 + \frac{\mu}{m_1} \sum_{s=-\infty}^{\infty} \frac{(1-\mu)^{|s|}}{\mu(2-\mu)} (k[s] - m_1\delta[s]) \\
&= 1 + \frac{k[0]-m_1}{m_1(2-\mu)} + \frac{2}{m_1(2-\mu)} \sum_{s=1}^{\infty} k[s](1-\mu)^s \\
&= 1 + \frac{k[0]-m_1}{m_1(2-\mu)} - \frac{2k[0]}{m_1(2-\mu)} + \frac{2}{m_1(2-\mu)} \sum_{s=0}^{\infty} k[s](1-\mu)^s \\
&= 1 - \frac{m_2 + m_1 - m_1^2}{m_1(2-\mu)} + \frac{2}{m_1(2-\mu)} k^*(1-\mu)
\end{aligned} \tag{A.22}$$

where $k^*(x)$ denotes the z-transform of $k[s]$.
To simplify the notation, let us introduce [50]

$$K[s] = \begin{cases} \frac{2}{m_1} k[s] & \text{if } s > 0 \\ \frac{1}{m_1} k[0] & \text{if } s = 0 \end{cases} \tag{A.23}$$

Then eq. (A.22) becomes

$$z_{\mathrm{geo}}(\mu) = 1 + \frac{K^*(1-\mu) - 1}{2-\mu} \tag{A.24}$$

where $K^*(\omega)$ is the z-transform of $K[s]$.

A.5 Peakedness and IDC

We investigate the connection between the peakedness function and the index of dispersion for counts (IDC, [17]). The IDC is a measure used to characterize the variability of an arrival stream on different time scales.

$$I[t] = \frac{V[t]}{E[t]} = \frac{V[t]}{m_1 t} \tag{A.25}$$

where $E[t]$ and $V[t]$ are the mean and variance of the number of arrivals in t consecutive epochs ($t = 1, 2, \ldots$).

First we express $I[t]$ by the covariance function. For this, first observe that the number of arrivals in an interval of length t can be expressed as the sum of the arrivals of t intervals of length 1, and the variance of the number of arrivals is therefore:

$$V[t] = tV[1] + 2\sum_{i=1}^{t-1}(t-i)k[i]. \tag{A.26}$$

Using the notation for $K[t]$ eq. (A.23), we get

$$I[t] = \frac{1}{t}\sum_{i=0}^{t-1}(t-i)K[i]. \tag{A.27}$$

Next we determine the z-transform of $I[t]$ (where we take $I[0] = 0$):

$$\begin{aligned} I^*(\omega) &= \sum_{t=1}^{\infty} I[t]\omega^t \\ &= \sum_{t=1}^{\infty}\sum_{i=0}^{t-1}\frac{1}{t}(t-i)K[i]\omega^t \\ &= \sum_{i=0}^{\infty}\sum_{t=i+1}^{\infty}\frac{1}{t}(t-i)K[i]\omega^t \\ &= \sum_{i=0}^{\infty}\sum_{t=i+1}^{\infty}K[i]\omega^t - \sum_{i=0}^{\infty}\sum_{t=i+1}^{\infty}\frac{i}{t}K[i]\omega^t. \end{aligned} \tag{A.28}$$

The first term of eq. (A.28) is

$$\begin{aligned} \sum_{i=0}^{\infty}\sum_{t=i+1}^{\infty}K[i]\omega^t &= \sum_{i=0}^{\infty}K[i]\omega^{i+1}\sum_{t=0}^{\infty}\omega^t \\ &= \sum_{i=0}^{\infty}K[i]\frac{\omega^{i+1}}{1-\omega} \\ &= \frac{\omega}{1-\omega}\sum_{i=0}^{\infty}K[i]\omega^i \\ &= \frac{\omega}{1-\omega}K^*(\omega) \end{aligned} \tag{A.29}$$

To evaluate the second term of eq. (A.28), note that

$$\sum_{t=i+1}^{\infty} \frac{\omega^t}{t} = \int_0^{\omega} \frac{x^i}{1-x} dx,$$

which can be proven by taking the derivative of both sides. Using this result,

$$\begin{aligned}
\sum_{i=0}^{\infty} \sum_{t=i+1}^{\infty} \frac{i}{t} K[i] \omega^t &= \sum_{i=0}^{\infty} iK[i] \sum_{t=i+1}^{\infty} \frac{\omega^t}{t} \\
&= \sum_{i=0}^{\infty} iK[i] \int_0^{\omega} \frac{x^i}{1-x} dx \\
&= \int_0^{\omega} \frac{1}{1-x} \sum_{i=0}^{\infty} iK[i] x^i dx \\
&= \int_0^{\omega} \left(\frac{1}{1-x} \frac{d}{dx} \sum_{i=0}^{\infty} K[i] x^{i+1} - \frac{1}{1-x} \sum_{i=0}^{\infty} K[i] x^i \right) dx \\
&= \left[\frac{1}{1-x} x K^*(x) \right]_0^{\omega} - \int_0^{\omega} \frac{1}{(1-x)^2} x K^*(x) dx - \int_0^{\omega} \frac{1}{1-x} K^*(x) dx \\
&= \frac{\omega}{1-\omega} K^*(\omega) - \int_0^{\omega} \frac{1}{(1-x)^2} K^*(x) dx \qquad \text{(A.30)}
\end{aligned}$$

using partial integration.

Using eq. (A.29) and eq. (A.30), the sum of the two terms is

$$I^*(\omega) = \int_0^{\omega} \frac{1}{(1-x)^2} K^*(x) dx. \qquad \text{(A.31)}$$

We can now express $K^*(\omega)$ by differentiating both sides:

$$K^*(\omega) = (1-\omega)^2 \frac{dI^*(\omega)}{d\omega}. \qquad \text{(A.32)}$$

We get the connection between the peakedness function and IDC by substituting this result into eq. (A.24):

$$z_{\text{geo}}(\mu) = 1 - \frac{\mu^2 \frac{dI^*(1-\mu)}{d\mu} + 1}{2-\mu}. \qquad \text{(A.33)}$$

This result can be used to get asymptotic results that connect the IDC and the peakedness. According to the final value theorem,

$$\lim_{t \to \infty} I[t] = \lim_{\omega \to 1} (1-\omega) I^*(\omega). \qquad \text{(A.34)}$$

If this limit is non-zero, then we can use the L'Hostipal rule:

$$\begin{aligned}
\lim_{t \to \infty} I[t] &= \lim_{\omega \to 1} \frac{I^*(\omega)}{1/(1-\omega)} \\
&= \lim_{\omega \to 1} (1-\omega)^2 \frac{dI^*(\omega)}{d\omega}
\end{aligned}$$

$$= \quad -\lim_{\mu \to 0} \mu^2 \frac{dI^*(1-\mu)}{d\mu} \tag{A.35}$$

Comparing it with eq. (A.33) and using this limiting value at $\mu = 0$, we get that if $\lim_{t\to\infty} I[t]$ is non-zero, then

$$z_{\text{geo}}(0) = \frac{\lim_{t\to\infty} I[t] + 1}{2}. \tag{A.36}$$

To see the connection between the first value, $I[1]$ and the peakedness function, we use the middle value theorem which says that

$$I[1] = \left.\frac{dI^*(\omega)}{d\omega}\right|_{\omega=0} = \left.\frac{dI^*(1-\mu)}{d\mu}\right|_{\mu=1} \tag{A.37}$$

Using eq. (A.33), we get that

$$z_{\text{geo}}(1) = I[1] = \frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}}. \tag{A.38}$$

A.6 Peakedness of Traffic Models

We now determine the peakedness function for geometrically distributed holding times for well-known and useful arrival processes.

A.6.1 Independent Identically Distributed Arrivals

The simplest case is when the numbers of arrivals in a slot are independent identically distributed with mean m_1 and second moment m_2.

In this case, $k[i] = 0$ (and also $K[i] = 0$) for all $i > 0$. Thus,

$$K^*(\omega) = K[0] = \frac{\text{Var}\{(w[i])\}}{\text{E}\{(w[i])\}} = \frac{m_2 - m_1^2}{m_1} \tag{A.39}$$

and

$$\begin{aligned} z_{\text{geo}}(\mu) &= 1 + \frac{K^*(1-\mu) - 1}{2-\mu} \\ &= 1 + \frac{\frac{\text{Var}\{(w[i])\}}{\text{E}\{(w[i])\}} - 1}{2-\mu} \end{aligned} \tag{A.40}$$

For the special case of Poisson arrivals, the distribution of arrivals in a slot is Poissonian. $\frac{\text{Var}\{w[i]\}}{\text{E}\{w[i]\}} = 1$ which gives $z_{\text{geo}}(\mu) = 1$. In the case of constant number of arrivals, $z_{\text{geo}}(\mu) = 1 - \frac{1}{2-\mu}$.

A.6.2 Markov Modulated Batch Bernoulli Process (MMBBP)

A very general Markovian arrival process is the Markov Modulated Batch Bernoulli Process or MMBBP. The modulating process is a discrete time Markov process. In each state of the modulating Markov-process, batch arrivals are generated independently according to a distribution corresponding to the state.

Let us denote by $\mathbf{P}$ and $\mathbf{D}$ the transition probability matrix and the row vector of steady-state distribution of the modulating Markov process, respectively. Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be diagonal matrices corresponding to the first and second moments of the number of arrivals in states. Let $\mathbf{e}$ be the column vector of all ones, $\mathbf{I}$ be the identity matrix.

We can express the mean number of arrivals as

$$m_1 = \mathbf{DM_1e} \tag{A.41}$$

and the second moment as

$$m_2 = \mathbf{DM_2e}. \tag{A.42}$$

The covariance function of the arrival process ($s > 0$) is written as

$$\begin{aligned} k[s] &= \mathrm{Cov}\{w[i], w[i+s]\} \\ &= \mathrm{E}\{w[i]w[i+s]\} - \mathrm{E}\{w[i]\}\,\mathrm{E}\{w[i+s]\} \\ &= \sum_{x,y} \mathrm{P}\{\mathcal{S}[i] = x, \mathcal{S}[i+s] = y\}\,(w[i]|\mathcal{S}[i] = x)(w[i+s]|\mathcal{S}[i+s] = y) - m_1^2 \\ &= \sum_{x} \mathrm{P}\{\mathcal{S}[i] = x\}\,(w[i]|\mathcal{S}[i] = x) \sum_{y} \mathrm{P}\{\mathcal{S}[i+s] = y|\mathcal{S}[i] = x\}\,(w[i+s]|\mathcal{S}[i+s] = y) - m_1^2 \\ &= \mathbf{DM_1P}^s\mathbf{M_1e} - m_1^2. \end{aligned} \tag{A.43}$$

where $\mathcal{S}[i] = x$ means that the Markov-chain is in state x at time i. For $s = 0$, $k[0] = \mathrm{Var}\{w[i]\} = m_2 - m_1^2$.

Using $k[s]$ we can determine its z-transform, $K(\omega)$ as follows:

$$\begin{aligned} K^*(\omega) &= \frac{1}{m_1}k[0] + \sum_{s=1}^{\infty} \frac{2}{m_1}k[s]\omega^s \\ &= \frac{m_2}{m_1} - m_1 + \frac{2}{m_1}\left(\sum_{s=1}^{\infty} \mathbf{DM_1P}^s\omega^s\mathbf{M_1e} - m_1^2\sum_{s=1}^{\infty}\omega^s\right) \\ &= \frac{1}{m_1}(2\omega\mathbf{DM_1P}(\mathbf{I} - \mathbf{P}\omega)^{-1}\mathbf{M_1}e + m_2) - m_1\frac{1+\omega}{1-\omega} \end{aligned} \tag{A.44}$$

This defines the peakedness function by eq. (A.24).

Using this result, we can obtain the peakedness function for special cases of the MMBBP process, such as the Batch Bernoulli Process (BBP), Markov Modulated Bernoulli Process (MMBP), Switched Batch Bernoulli Process (SBBP), as described in Section 2.2.4. For the discrete time batch renewal process we can also obtain the peakedness based on the derivation of the covariance function in [50], as described in Section 2.2.4.

Bibliography

[1] A. Bakre and B. R. Badrinath. I-TCP: Indirect TCP for mobile hosts. In *Proc. 15th International Conference on Distributed Computing Systems (ICDCS)*, May 1995.

[2] B. Bakshi, P. Krishna, N. H. Vaidya, and D. K. Pradhan. Improving performance of TCP over wireless networks. In *17th Inernational Conference on Distributed Computing Systems*, Baltimore, May 1997.

[3] Hari Balakrishnan, Venkata N. Padmanabhan, Srinivasan Seshan, and Randy H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5:756–769, Dec 1997.

[4] Hari Balakrishnan, Srinivasan Seshan, and Randy H. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, 1(4), December 1995.

[5] Jon C. R. Bennett and Hui Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689.

[6] P. Bhagwat, Partha Bhattacharya, Arvind Krishna, and Satish K. Tripathi. Enhancing throughput over wireless lans using channel state dependent packet scheduling. In *Proceedings of INFOCOM*, April 1996.

[7] G. Bianchi and A. T. Campbell. A programmable MAC. In *Proceedings of IEEE International Conference on Universal Personal Communicatons (ICUP)*, Florence, Italy, October 1998.

[8] Giuseppe Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, March 2000.

[9] Giuseppe Bianchi, Luigi Fratta, and Matteo Oliveri. Performance evaluation and enhancement of the CSMA/CA MAC protocol for 802.11 wireless LANs. In *Proc. PIMRC*, Oct 1996.

[10] Bluetooth specification 1.1.

[11] Bluetooth SIG PAN working group.

[12] Broadband radio access networks (bran) HIPERLAN type 2 data link control (dlc) layer part 1: Basic data transport functions, April 2000.

[13] Kevin Brown and Suresh Singh. M-TCP: TCP for mobile cellular networks. *ACM Computer Communication Review*, 27(5), 1997.

[14] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13, June 1995.

[15] Charles Chien, Mani B. Srivastava, Rajeev Jain, Paul Lettieri, Vipin Aggarwal, and Robert Sternowski. Adaptive radio for multimedia wireless links. *IEEE JSAC*, May 1999.

[16] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations, 1999. RFC2501.

[17] D. R. Cox and P. A. W. Lewis. *The Statistical Analysis of Series of Events*. Methuen & Co Ltd, 1966.

[18] A. DeSimone, M. C. Chuah, and O. C. Yue. Throughput performance of transport-layer protocols over wireless lans. In *Proceedings of Globecom*, 1993.

[19] A. E. Eckberg, Jr. Generalized peakedness of teletraffic processes. In *ITC-10*, Montreal, 1983.

[20] A. E. Eckberg, Jr. Approximations for bursty (and smoothed) arrival queueing delays based on generalized peakedness. In *ITC-11*, Kyoto, Japan, 1985.

[21] D. Eckhardt and P. Steenkiste. Improving wireless LAN performance via adaptive local error control. In *Proceedings of INCP*, 1998.

[22] D. Eckhardt and P. Steenkiste. Effort-limited fair (ELF) scheduling for wireless networks. In *Proceedings of IEEE INFOCOM*, 2000.

[23] David Eckhardt and Peter Steenkiste. A trace-based evaluation of adaptive error correction for a wireless local area network. *Mobile Networks and Applications*, 4(4):273–287, 1999.

[24] Jon Postel (editor). Transmission control protocol, 1981. RFC793.

[25] A. Ephremides, J. Wieselthier, and D. Baker. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proceedings of the IEEE*, 1987.

[26] Phil Karn et al. Advice for internet subnetwork designers, July 2001. Internet Draft of the PILC working group of the IETF, work in progress.

[27] S. Blake et al. An architecture for differentiated services, 1998. RFC2475.

[28] ETSI Broadband Radio Access Networks project. http://www.etsi.fr/bran/.

[29] K. Fall and S. Floyd. Simulation-based comparisons of Tahoe, Reno, and SACK TCP. *Computer Communications Review*, 26(3):5–21, July 1996.

[30] Gorry Fiarhurst and Lloyd Wood. Link ARQ issued for IP traffic, March 2001. Internet Draft of the PILC working group of the IETF, work in progress.

[31] Sally Floyd and Van Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4), August 1995.

[32] Christine Fragouli, Vijay Sivaraman, and Mani B. Srivastava. Controlled multimedia wireless link sharing via enhanced class-based queuing with channel-state-dependent packet scheduling. In *IEEE INFOCOM'98*, March 1998.

[33] Jim Geier. *Wireless LANs: Implementing Interoperable Networks*. Macmillan Technical Publishing, first edition, 1999.

[34] J. Gomez, A. T. Campbell, and H. Morikawa. A systems approach to prediction, compensation and adaptation in wireless networks. In *First ACM/IEEE International Workshop on Wireless and Mobile Multimedia (WoWMoM'98)*, Dallas, 1998.

[35] Pawan Goyal, Harrick M. Vin, and Haichen Cheng. Start-time fair queuing: A scheduling algorithm for integrated services packet switching networks. *IEEE/ACM Transactions on Networking*, 5(5):690–704, October 1997.

[36] R. Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *IEEE Journal on Selected Areas in Communications*, 9(2), February 1991.

[37] HiperLAN/2 Global Forum. http://www.hiperlan2.com/.

[38] Marko Hännikäinen, Timo D. Hämäläinen, Markku Niemi, and Jukka Saarinen. Trends in personal wireless data communications. *Computer Communications*, 25(1), 2002.

[39] Zs. Haraszti, I. Dahlquist, A. Faragó, and T. Henk. PLASMA - an integrated tool for ATM network operation. In *International Switching Symposium ISS '95*, Berlin, Germany, April 1995.

[40] H. Heffes and J. M. Holtzman. Peakedness of traffic carried by a finite trunk group with renewal input. *The Bell System Technical Journal*, 52(9):1617–1642, November 1973.

[41] HIPERLAN type 2; data link control (DLC) layer; part 1: Basic data transport functions, 2002. ETSI TS 101 761-2 .

[42] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A rate-adaptive MAC protocol for wireless networks. In *Proceedings of MOBICOM*, 2001.

[43] IEEE. IEEE 802.11 family of standards.

[44] Internet engineering task force. http://www.ietf.org.

[45] Martin Johnsson. HiperLAN/2 — the broadband radio transmission technology operating in the 5ghz frequency band, 1999. White Paper of the HIPERLAN/2 Global Forum.

[46] Jamshid Khun-Jush, Gran Malmgren, Peter Schramm, and Johan Torsner. Overview and performance of HIPERLAN type 2 - a standard for broadband wireless communications. In *Proceedings of VTC2000 Spring*, Tokyo, Japan.

[47] Jamshid Khun-Jush, Gran Malmgren, Peter Schramm, and Johan Torsner. HIPERLAN type 2 for broadband wireless communication. *Ericsson Review*, 2000.

[48] Jamshid Khun-Jush, Peter Schramm, Udo Wachsmann, and Fabian Wenger. Structure and performance of the HIPERLAN/2 physical layer. In *Proceedings of VTC2000 Fall*, Boston, USA.

[49] Leonard Kleinrock. *Queueing Systems. Vol I: Theory.* John Wiley & Sons, 1975.

[50] Demetres Kouvatsos and Rod Fretwell. Closed form performance distributions of a discrete time $gi^g/d/1/n$ queue with correlated traffic. In R. Onvural and S. Folida, editors, *Proceedings of the 6th IFIP Conference on Performance of Computer Networks*, Istambul, October 1995. Chapman and Hall Publishers, London.

[51] Demetres Kouvatsos and Rod Fretwell. Batch renewal process: Exact model of traffic correlation. In *High Speed Networking for Multimedia Application*, pages 285–304. Kluwer Academic Press, 1996.

[52] Ching Law, Amar K. Mehta, and Kai-Yeung Siu. Performance of a new Bluetooth scatternet formation protocol. In *Mobihoc*, 2001.

[53] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel Wilson. On the self-similar nature of Ethernet traffic (extended version). *IEEE/ACM Transactions on Networking*, 2(1), February 1994.

[54] P. Lettieri and M. Srivastava. Adaptive frame length control for improving wireless link throughput, range, and energy efficiency. In *Proceedings of IEEE Infocom*, San Francisco, USA, March 1998.

[55] Songwu Lu, Vaduvur Bharghavan, and Rayadurgam Srikant. Fair shceduling in wireless packet networks. In *SIGCOMM'97.*

[56] Songwu Lu, Thyagarajan Nanadagopal, and Vaduvur Bharghavan. A wireless fair service algorithm for packet cellular networks. In *ACM MOBICOM'98*, August 1998.

[57] Reiner Ludwig. A case for flow adaptive wireless links. Technical Report UCB//CSD-99-1053, University of California at Berkeley, May 1999.

[58] Reiner Ludwig and Randy H. Katz. The Eifel algorithm: Making TCP robust against spurious retransmissions. *ACM Computer Communications Review*, January 2000.

[59] Reiner Ludwig, Almudena Konrad, Anthony D. Joseph, and Randy H. Katz. Optimizing the end-to-end performance of reliable flows over wireless links. *Wireless Networks Journal*, 1999.

[60] Reiner Ludwig, Bela Rathonyi, Almudena Konrad, Kimberly Oden, and Anthony Joseph. Multi-layer tracing of tcp over a reliable wireless link. In *Proceedings of SIGMETRICS*, 1999.

[61] Joseph Macker and Scott Corson (chairmen). MANET (Mobile Ad Hoc Networking) working group of the IETF.

[62] Brian L. Mark, David L. Jagerman, and G. Ramamurthy. Application of peakedness measures to resource allocation in high-speed networks. In *Proceedings of ITC-15, Washington D.C., USA*, June 1997.

[63] Brian L. Mark, David L. Jagerman, and G. Ramamurthy. Peakedness measures for traffic characterization in high-speed networks. In *Proceedings of IEEE INFOCOM'97*, 1997.

[64] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options, 1996. RFC 2018.

[65] M. Meyer. TCP performance over GPRS. In *Proceedings of IEEE WCNC*, 1999.

[66] Sándor Molnár. *Evaluation of Quality of Service and Network Performance in ATM Networks*. PhD thesis, Technical University of Budapest, Department of Telecommunications and Telematics, 1995.

[67] Sándor Molnár, István Cselényi, and Nils Björkman. ATM traffic characterization and modeling based on the leaky bucket algorithm. In *IEEE Singapore International Conference on Communication Systems*, Singapore, November 1996.

[68] Sándor Molnár and Attila Vidács. On modeling and shaping self-similar ATM traffic. In *Proceedings of ITC-15, Washington D.C., USA*, June 1997.

[69] Thyagarajan Nanadagopal, Songwu Lu, and Vaduvur Bharghavan. A unified architecture for the design and evaluation of wireless fair queueing algorithms. In *ACM MOBICOM'99*, August 1999.

[70] T. S. Eugene Ng, Ion Stoica, and Hui Zhang. Packet fair queueing algorithms for wireless networks with location-dependent errors. In *INFOCOM'98*.

[71] A. Ohta, M. Yoshioka, T. Sugiyama, and M. Umehira. PRIME ARQ: A novel ARQ scheme for high-speed wireless ATM design, implementation and performance evaluation. In *Proc. VTC 98*, Ottawa, Canada.

[72] Raif O. Onvural. *Asynchronous Transfer Mode Networks, Performance Issues*. Artech House, Boston, London, 1994.

[73] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Transactions on Networking*, 1:344–357, June 1993.

[74] Vern Paxson and Sally Floyd. Wide area traffic: The failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[75] A. Pursley. The role of spread spectrum in packet radio networks. *Proceedings of the IEEE*, 75(1), 1987.

[76] Parameswaran Ramanathan and Prathima Agrawal. Adapting packet fair queueing algorithms to wireless networks. In *Proceedings of MOBICOM*, pages 1–9, 1998.

[77] Theodore S. Rappaport. *Wireless Communications*. Prentice Hall, 1996.

[78] Miklós Aurél Rónai and Eszter Kail. A simple neighbour discovery procedure for Bluetooth ad hoc networks. In *Proceedings of Globecom*, December 2003.

[79] Oliver Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. In *Proceedings of the 20th Annual Conference on Local Computer Networks*, pages 397–406, Minneapolis, MN, 1995. ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG/.

[80] Peter Sági. Investigations on the resource requesting algorithms of a wireless local area network. Master's thesis, Budapest University of Technology and Economics, Department of Telecommunications and Telematics, 2000.

[81] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire. Distributed topology construction of Bluetooth personal area networks. In *Proceedings of Infocom*, 2001.

[82] N.K.G. Samaraweera and G. Fairhurst. Reinforcement of TCP error recovery for wireless communication. *Computer Communication Review*, 28(2), April 1998.

[83] Kotikalapudi Sriram and Ward Whitt. Characterizing superposition arrival processes in packet multiplexers fo voice and data. *IEEE Journal on Selected Areas in Communications*, 4(6), September 1986.

[84] G. D. Stamoulis, M. E. Anagnostou, and A. D. Georgantas. Traffic source models for ATM networks: a survey. *Computer Communications*, 17(6), 1994.

[85] W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[86] Andrew S. Tanenbaum. *Computer Networks*. Prentice-Hall International, third edition, 1996.

[87] Zhenyu Tang and J. J. Garcia-Luna-Aceves. Hop-reservation multiple access for multichannel packet radio networks. *Computer Communications*, 23(10), 2000.

[88] Johan Torsner and Göran Malmgren. Radio network solutions for HIPERLAN/2. In *Proceedings of VTC1999 Spring*, Houston.

[89] Jean Tourrilhes. Dwell adaptive fragmentation: how to cope with short dwells required by multimedia wireless LANs. In *Proceedings of Globecom*, San Francisco, 2000.

[90] Jean Tourrilhes. Fragment adaptive reduction: Coping with various interferers in radio unlicensed bands. In *Proceedings of ICC*, Helsinki, Finland, 2001.

[91] N. Vaidya, M. Mehta, C. Perkins, and G. Montenegro. Delayed duplicate acknowledgements: A TCP-unaware approach to improve performance of TCP over wireless. *Technical Report, Computer Science Dept., Texas A&M University*, 1999.

[92] András Gergely Valkó. *Design and Analysis of Cellular Mobile Data Networks*. PhD thesis, Budapest University of Technology and Economics, Department of Telecommunications and Telematics, 1999.

[93] R. Yavatkar and N. Bhagwat. Improving end-to-end performance of TCP over mobile internetworks. In *Proceedings of Mobile 94 Workshop on Mobile Computing Systems and Applications*, December 1994.

[94] Gergely V. Záruba, Stefano Basagni, and Imrich Chlamtac. Bluetrees – scatternet formation to enable Bluetooth-based ad hoc networks. In *Proceedings of ICC*, 2001.

[95] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83(10), October 1995.

[96] Eustathia Ziouva and Theodore Antonakopoulos. CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, 25(3), 2002.

[97] M. Zorzi, R.R. Rao, and L.B. Milstein. Error statistics in data transmission over fading channels. *IEEE Transactions on Communications*, Nov 1998.

[98] S. Zürbes, W. Stahl, K. Matheus, and J. Haartsen. Radio network performance of Bluetooth. In *Proceedings of ICC*, 2000.

Publications

[J] **Journal Papers**

[J1] Gy. Miklós, "Bluetooth: Olcsó, drótnélküli helyi összeköttetés (Bluetooth: cheap wireless local connectivity)," Magyar Távközlés, vol. 11, No. 9, pp. 11-14, September 2000 (in Hungarian).

[J2] Gy. Miklós, F. Kubinszky, A. Rácz, Z. R. Turányi, A. G. Valkó, M. A. Rónai, S. Molnár, "A Novel Scheme to Interconnect Multiple Frequency Hopping Channels into an Ad Hoc Network," ACM Mobile Computing and Communications Review (MC2R), Special Issue on Wireless PAN & Sensor Networks, vol. 8, No. 1, pp. 109-124, January 2004.

[C] **Conference and Workshop Papers**

[C1] S. Molnár, Gy. Miklós, "On Burst and Correlation Structure of Teletraffic Models," 5th IFIP Workshop on Performance Modelling and Evaluation of ATM Networks, 21-23 July 1997, Ilkley, U.K.

[C2] S. Molnár, Gy. Miklós, "Peakedness Characterization in Teletraffic," IFIP TC6, WG6.3 conference, Performance of Information and Communications Systems, PICS'98 Lund, Sweden, May 25-28, 1998.

[C3] Gy. Miklós, S. Molnár, "Fair Bandwidth Allocation of a Wireless Base Station," First Workshop on IP Quality of Service for Wireless and Mobile Networks (IQWiM99), April 1999, Aachen, Germany.

[C4] Gy. Miklós, S. Molnár "Fair Allocation of Elastic Traffic for a Wireless Base Station," IEEE GLOBECOM'99, December, 1999 Rio de Janeiro, Brazil.

[C5] H. Li, J. Lindskog, G. Malmgren, Gy. Miklós, F. Nilsson, G. Rydnell, "Automatic Repeat Request (ARQ) Mechanism in HIPERLAN/2," VTC2000 Spring, May 2000, Tokyo, Japan.

[C6] Gy. Miklós, S. Molnár "Distributed Fair Bandwidth Allocation of a Wireless Base Station," Networking 2000, May 14-19, 2000, Paris, France. In: G. Pujolle et al. (Eds.), "Lecture Notes in Computer Science 1815," Springer-Verlag Berlin Heidelberg 2000, pp 689-701.

[C7] Gy. Miklós, S. Molnár, "Analysis of a Distributed Wireless Fair Scheduling Scheme," ITC Specialist Seminar on Mobile Systems and Mobility, pp 301-312, March, 2000, Lillehammer, Norway.

[C8] Gy. Miklós, A. Rácz, Z. Turányi, A. Valkó, P. Johansson "Performance Aspects of Bluetooth Scatternet Formation," Proceedings of MobiHoc 2000, pp. 147-148, Boston, USA.

[C9] A. Rácz, Gy. Miklós, F. Kubinszky, A. Valkó, "A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets," Proceedings of MobiHoc 2001, pp. 193-203, Long Beach, USA.

[T] **Technical Reports**

[T1] Gy. Miklós, "Peakedness Measures," technical report, High Speed Networks Laboratory, Technical University of Budapest, 1997.

[T2] S. Molnár, Gy. Miklós, "Generalized Peakedness in Discrete Time," COST 257 TD(98)052, Granada, Spain, September 24-25, 1998.

[T3] S. Molnár, Gy. Miklós, "A Simple Compensation Mechanism for Wireless Resource Allocation," COST 257 TD(99)037, September 30 - October 1, 1999, Larnaca, Cyprus.

[S] **Standardisation contributions**

[S1] Ericsson, "Selective repeat ARQ mechanism in HiperLAN2," HL12.5ERI7a, March 1999, Stockholm, Sweden, contribution to ETSI BRAN standardisation.

[S2] Ericsson, "ARQ message formats," HL13ERI8a, April 1999, Stockholm, Sweden, contribution to ETSI BRAN standardisation.

[S3] Ericsson, "Flow control in HiperLAN 2 ARQ protocol," HL14ERI11a, June 1999, Sophia Antipolis, France, contribution to ETSI BRAN standardisation.

[S4] Ericsson, "Polling bit in FCCH RG for RR," HL14ERI20a, June 1999, Sophia Antipolis, France, contribution to ETSI BRAN standardisation.

[S5] Ericsson, "Discard mechanism in the HiperLAN2 ARQ protocol," HL14ERI22b, June 1999, Sophia Antipolis, France, contribution to ETSI BRAN standardisation.

[S6] Ericsson, "Error handling in the HiperLAN2 ARQ protocol," HL14ERI24a, June 1999, Sophia Antipolis, France, contribution to ETSI BRAN standardisation.

[P] **Patents**

[P1] Gy. Miklós, "Discard Mechanism for Selective Repeat Automatic Repeat Request," US patent application 09/273259, March 1999.

[P2] J. Lindskog, J. Ollson, G. Malmgren, Gy. Miklós, "Ordering of LCH&SCH," US patent application 09/544219, April 2000.

[P3] G. Rydnell, J. Lindskog, G. Malmgren, Gy. Miklós, H. Li, F. Nilsson, "ARQ C-PDU Format," US patent application 60/128041, April 2000.

[P4] Gy. Miklós, M. Félegyházi, "Traffic Dependent Bluetooth Scatternet Optimization Procedure," US patent application No. 09/666529, September 2000.

[P5] Gy. Miklós, Z. Turányi, A. G. Valkó, "Flexible Neighbour Discovery in Bluetooth," UK patent application No. 0110397.7, April 2001.

[P6] Gy. Miklós, Z. Turányi, A. G. Valkó, "Lightweight Piconet Concept in Bluetooth," UK patent application No. 0110396.9, April 2001.

[P7] Gy. Miklós, Z. Turányi, A. G. Valkó, F. Kubinszky, A. Rácz "Random Access Communication in Bluetooth," UK patent application No. 0110399.3, April 2001.

[P8] A. Rácz, Gy. Miklós, A. G. Valkó, F. Kubinszky, "Signaling Free, Self Learning Scatternet Scheduling Using Checkpoints," US patent application No. 09/840241, April 2001.

[P9] Gy. Miklós, A. Rácz, "Predictable Communication Establishment in Ad-hoc Wireless Network" US patent application No. 10/039606, October 2001.

[P10] Gy. Miklós, A. Rácz, "Fast Hard Handover" WIPO patent application No. PCT/SE2004/000192, February 2004.