Concurrent Access and Traffic Control Methods in Wireless Communication Networks

G.J. Hoekstra
Concurrent Access and Traffic Control Methods
in Wireless Communication Networks
ISBN: 978-94-6191-179-7
NUR: 919

The research in this thesis has been carried out in the context of the Casimir project: Analysis of Distribution Strategies for Concurrent Access in Wireless Communication Networks, registered under file number 018.002.012.

This Casimir project was supported by Thales Nederland B.V., Centrum Wiskunde & Informatica (CWI), Alcatel-Lucent Nederland B.V. and by the Netherlands Organisation for Scientific Research (NWO).

Cover designed by Christine van Rees.

Printed by Ipskamp Drukkers B.V., The Netherlands.

VRIJE UNIVERSITEIT

Concurrent Access and Traffic Control Methods in Wireless Communication Networks

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van Doctor
aan de Vrije Universiteit Amsterdam,
op gezag van de Rector Magnificus
prof.dr. L.M. Bouter,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen
op 9 maart 2012 om 11:45 uur
in de xxx,
De Boelelaan 1105

door

Geert Jan Hoekstra

geboren op 16 mei 1973 te Ooststellingwerf

promotor:

prof.dr. R.D. van der Mei

manuscriptcommissie:

prof.dr. J.L. van den Berg	TNO en Universiteit Twente
dr. S. Bhulai	Vrije Universiteit Amsterdam
prof.dr. C. Blondia	Universiteit Antwerpen, België
prof.dr.ir. S.C. Borst	Bell Labs, Alcatel-Lucent, U.S.A. en Technische Universiteit Eindhoven
prof.dr.ir. B.R.H.M. Haverkort	Universiteit Twente
dr.ir. H.B. Meeuwissen	TNO

Contents

1 Introduction **1**
- 1.1 Background and motivation . . . 1
- 1.2 Literature . . . 2
- 1.3 Overview of the thesis . . . 4
- 1.4 Publications . . . 7

2 Concept of Effective Service Time in Wireless Networks **9**
- 2.1 Introduction . . . 10
- 2.2 Modeling . . . 12
- 2.3 Model validation by simulation . . . 28
- 2.4 Model validation by testbed experiments . . . 34
- 2.5 Engineering guidelines for practical application . . . 37
- 2.6 Conclusions and further research . . . 41

3 Tail-Optimal Static Traffic Splitting over Multiple Networks **43**
- 3.1 Introduction . . . 44
- 3.2 Problem formulation . . . 45
- 3.3 Heuristic derivation of the proposed splitting rule . . . 47
- 3.4 Reduced load equivalence . . . 49
- 3.5 Simulation results . . . 51
- 3.6 Tail behavior versus mean behavior . . . 55
- 3.7 Conclusions and further research . . . 57

4 Mean-Optimal Static Traffic Splitting over Multiple Networks **59**
- 4.1 Introduction . . . 59
- 4.2 Model description . . . 60
- 4.3 Analysis . . . 62
- 4.4 Numerical results . . . 69
- 4.5 Conclusions and further research . . . 77

5 Dynamic Traffic Assignment to Multiple Networks **79**
- 5.1 Introduction . . . 79
- 5.2 Model description and analysis . . . 82
- 5.3 Numerical experiments . . . 86
- 5.4 Conclusions and further research . . . 95

6 Dynamic Traffic Splitting over Multiple Networks 97
6.1 Introduction . . . 97
6.2 Concurrent access network model . . . 100
6.3 Analysis of the CAN-model . . . 101
6.4 Score-function based splitting method . . . 105
6.5 Experimental validation of score-function based approach . . . 106
6.6 Comparison between Concurrent Access strategies . . . 111
6.7 Conclusions and further research . . . 120

7 Traffic Control in Wireless Networks 123
7.1 Introduction . . . 124
7.2 Multi-service traffic profiles . . . 126
7.3 Guaranteeing QoS for WLAN networks . . . 137
7.4 Lessons learned from implementing multi-service traffic profiles . 143
7.5 Conclusions and further research . . . 147

8 On Regulating Traffic in Shared Wireless Access Media 151
8.1 Introduction . . . 152
8.2 Mechanism for policing traffic in shared media . . . 153
8.3 Performance evaluation . . . 158
8.4 Conclusions and further research . . . 173

Appendix A 175

Appendix B 177

Appendix C 183

Appendix D 189

Publications of the author 195

Samenvatting 199

Bibliography 204

Chapter 1

Introduction

Over the past two decades the use of wireless communication networks has grown at an unprecedented rate, and this growth is not likely to come to an end in the near future. The main barrier to the sustained growth is the limited capacity of radio resources that may fluctuate depending on the propagation environment. In practical deployments these scarce resources have to be shared among a multitude of users, each of which shows random behavior in terms of application usage and mobility. In the competitive markets of wireless communication services it is essential for network operators to deliver high-quality services at sharp prices. This raises the need for smart and distinguishing methods to utilize and control the scarce wireless network resources in a cost-efficient manner.

Many geographical locations are covered by a multitude of overlapping wireless networks. This phenomenon, throughout referred to as *Concurrent Access* (CA), has opened up tremendous possibilities for increasing capacity, improving robustness, and enhancing Quality of Service (QoS). Despite the enormous potential for quality improvement, only little is known about how to fully exploit the possibilities offered by CA. In this context, the goal of this thesis is to evaluate and optimize the efficiency of different CA approaches and traffic control mechanisms.

1.1 Background and motivation

In traditional tactical or telecommunication networks, services are tightly coupled with the underlying transmission medium. For example, in traditional networks classical voice telephony services are offered via the circuit-switched telephone networks, television services are offered via television broadcasting networks and data services (e.g., financial transactions) are offered via dedicated data networks. With the emergence of the Internet Protocol (IP) this 'stovepipe' model has evolved into a model in which services and their transmission medium are no longer strictly coupled. In fact, in most contemporary

communication systems a variety of services is transported by a plurality of network types and technologies.

At the same time spectacular advances in networking technology have been realized over the last few years, leading to a significant increase of the available capacity in both wired and wireless networks. The possibilities offered by modern wireless networking technologies have boosted the demand for high-capacity wireless services. This, in turn, has led to the introduction of new, and highly-efficient, network technologies that achieve high peak data rates. Consequently, many contemporary wireless network technologies have closely approached the Shannon limit on channel capacity, leaving complex signal processing techniques room for only modest improvements in the data transmission rate [34].

A promising solution to satisfy the increasing demand for high application data rates is the concurrent use of multiple wireless networks. On multi-homed devices, for example, the data rate available to the applications may strongly benefit from the overlapping coverage of a wide range of wireless access technologies that operate in different frequency bands and already achieve very high spectral efficiencies. The approach to benefit from accessing multiple networks simultaneously, called CA, potentially delivers significant advantages, including enhanced performance improvement and robustness. Today, however, little is known about how to effectively exploit this enormous potential by smartly splitting traffic streams over multiple concurrent wireless networks. This is the motivation for the research pursued in this thesis.

1.2 Literature

In the context of communication systems the concurrent use of multiple network resources in parallel was already described for a Public Switched Digital Network (PSDN) [36]. Here inverse multiplexing was proposed as a technique to perform the aggregation of multiple independent information channels across a network to create a single higher-rate information channel. Many different forms of parallelism occur throughout different protocol layers in communication systems to enhance reliability, e.g., protecting working channels that transfer voice signaling using the Stream Control Transmission Protocol (SCTP) [111], or to increase network performance for Wireless LANs (WLANs) using multiple antennas in the IEEE 802.11n standard [6].

Many research efforts are focused on combining the capacity of multiple networks concentrate on the link layer, the transport layer and on the application layer of communication systems. At the *data link layer*, approaches have been proposed for switching between several homogeneous networks [31] and scheduling over heterogeneous networks [72, 18]. However, approaches at this layer require modifications for each different network interface that needs to be supported and, moreover, switching the data segments of the same Transport

Control Protocol (TCP) session over different network links, even in homogeneous networks, adversely affects TCP performance [31, 58]. At the *transport layer*, two main areas can be distinguished: one concentrating on the use of SCTP (e.g., see [61, 60, 59]) and the other on using or modifying TCP. Within the IETF, SCTP has evolved from a transport protocol for voice-signaling traffic into one that allows various types of information to be transported over different network paths. It needs to be pointed out that the functionality for efficiently using concurrent paths is not considered by the standard, meaning that distributing and re-sequencing the data should be implemented separately, and that the flow- and congestion control mechanism is the same for the possibly different networks used in parallel, which is not in the interest of overall efficient link utilization nor application performance [58, 124]. Several proposals to modify, use and extend TCP for exploiting multiple networks have appeared. The most prominent being a modification of TCP, called pTCP [58], later followed by mTCP [124] that both aimed at enhancing TCP to be capable of using multiple networks concurrently. Finally, at the *application layer*, another approach can be applied to establish multiple connections to one or more endpoints for transferring information in a distributed manner, as is done by Peer-to-Peer (P2P) techniques. In this thesis we concentrate on the transport layer for a practical realization of CA, because it preserves existing applications, abstracts from network particularities and seems the most promising layer to operate on with respect to achieving high performance gains.

In the literature, many models have been proposed that can predict the performance of several networks separately. Most prominently, the application performance of file transfers is often modeled by Processor Sharing (PS)-based models [22, 76, 120] that have shown to be applicable to a wide variety of wireless access networks, including CDMA 1xEV-DO, WLAN, and UMTS-HSDPA. In fact, when proper parameterization is applied, PS models may accurately predict the performance of file transfers over WLANs [51] by taking into account the complex dynamics of the application and its underlying protocol-stack. PS models provide an abstraction from the highly complicated packet-level details of a network that reduces the model complexity significantly. By accounting for those essential factors that determine the network performance, highly accurate model predictions can be obtained, while an exact mathematical analysis of the model is still feasible.

Despite the applicability of PS-based models to real communication networks, little is known on PS-based models suitable for modeling the use of multiple networks concurrently. In a queueing-theoretical context, the distribution and re-assembly of tasks into subtasks are typically modeled by fork-join constructions [71]. In cases where the processing times of the subtasks are independent, exact or numerical analysis is relatively simple (e.g., [32]), whereas the inclusion of dependent processing times (e.g., due to queueing or job splitting) typically leads to very complex analysis (e.g., [37, 81]) and no closed-form solution exists. In [75] and [74], the author analyzes a similar model but with FCFS queues

and with probabilistic splitting. We further refer to Altman et al. [11], who consider routing policies in a distributed versus centralized environment. In general our queueing model falls within the framework of fork-join queueing networks, see [12] for an extensive overview.

For PS-queues that process the tasks of a job in parallel, the complex correlation structure between the sojourn times at the PS-queues makes an exact detailed mathematical analysis of the model impossible. As a result, the available literature on queueing models with regard to traffic-splitting is not widely adopted and hence leaves a gap between theory and practice. As an exception, Key et al. investigate the efficiency of combining multipath routing and congestion control in TCP-based networks. In [68] they show that under certain conditions the allocation of flows to paths is optimal and independent to the flow control algorithm used. In [69] it is shown that with RTT bias uncoordinated control can lead to inefficient equilibria, while without RTT bias, both coordinated and uncoordinated Nash equilibria correspond to desirable welfare maximizing states.

The literature leaves a clear gap between theory and practice: the existing theory provides important fundamental insight, but the models often rely on simplifying assumptions that cannot be met by practical deployments. On the practical side, the research efforts on splitting algorithms and implementations were concentrated at increasing the throughput on multiple networks (often in the absence of background traffic) without having a notion of the user-level performance in the presence of background traffic. In fact, the impact that CA approaches in networks have on the performance of other traffic is a subject that is not well-known in literature.

In this thesis a three-stage approach is applied. First, a network model is devised that accurately describes the application-level performance by properly parameterizing a queueing abstraction. Second, several CA approaches are considered in the context of an abstract queueing system. By doing so, the performance that various CA approaches achieved in a queueing system can be described, optimized and mutually compared. The third, most essential, stage closes the circle by applying the results from the queueing models to simulation environments and testbed networks to demonstrate their practical impact.

1.3 Overview of the thesis

In Chapter 2 we introduce a new concept for modeling data traffic flows in a communication network by jobs that are processed by a Processor Sharing (PS) queue. The influence of complex combined dynamics and protocol overhead of multiple communication layers on data traffic flows in real networks can be accounted for in an explicit expression for a single parameter which will be called the *effective service time*. Based on the effective service time, the *effective load*

is defined to describe the performance of data flows in a network with an $M/G/1$ PS model. Extensive validation by means of simulations and experiments conducted with network equipment show that accurate predictions can be obtained from our model for a wide range of parameter combinations. Additionally, our model can be used to parameterize a PS-based queueing network, such that the outcomes correspond to the performance of real networks or vice versa.

In Chapter 3 we consider a CA network that consists of parallel communication networks, each of which is modeled as a PS-queue that handles two types of traffic: foreground and background streams. The foreground traffic stream consists of jobs, each of which is split into fragments according to a *fixed splitting rule*. These fragments are processed by the parallel PS-queues. Upon completion of all fragments, the job is re-assembled at the receiving end. The background streams use dedicated queues without being split. By applying a fixed job splitting rule, the approach is referred to as *static job splitting*. The aim of static job splitting of files and transferring the portions through parallel networks is to reduce the transfer time of a foreground file transfer. Based on the model for effective service time from Chapter 2 the network may be represented by PS-queues and the files by jobs to study a file-splitting rule in a queueing theoretical context. The corresponding queueing network is called a CA job-split queueing network that consists of parallel PS-queues. Based on a Reduced Load Approximation (RLA) it is proven that a simple splitting rule achieves tail optimality with respect to the sojourn time of jobs that are split. Extensive simulations demonstrate that this simple rule indeed performs well, not only with respect to the tail asymptotics, but also with respect to the mean sojourn times.

In Chapter 4 the CA job-split queueing network from Chapter 3 is analyzed with the aim to minimize the mean sojourn time by using static splitting. As a result of static splitting, the sojourn times of a job's fragments in the different PS-queues are generally correlated, which prohibits an exact analysis. Therefore, a new approximation is developed that combines light- and heavy-traffic asymptotics, which then leads to an approximation for the optimal splitting rule with respect to the mean sojourn time of foreground jobs. Extensive simulations demonstrate that the differences between the approximated optimal splitting rule and the estimated optimum with respect to the expected foreground sojourn time are extremely small for a wide range of the parameter settings.

In Chapter 5 a CA network of parallel communication networks is considered in which the files from the foreground traffic are assigned to one of the available networks in the presence of background traffic streams in each of the parallel networks. The assignment of foreground files to one of the available networks is characterized in the context of this thesis by the term *dynamic job assignment*. Similar to the CA job-split optimization problem, the dynamic assignment problem in this chapter is studied in a queueing theoretical context using a CA job-assignment model, where the networks are represented by PS-queues

and the files by jobs that are served by these queues. The goal in this chapter is to develop a dynamic policy that minimizes the mean sojourn time of the foreground traffic. A full-information state policy is developed by solving a Markov decision problem that requires information on the number of foreground and background jobs in each queue. It is realized that the information that is available to the decision maker does not allow to distinguish the number of foreground and background jobs in the network, but instead *only* has partial information on the *total* number of jobs. To this end, we develop and evaluate a Bayesian learning algorithm that splits a stream by optimally assigning entire jobs to different queues. The optimality of the partial information algorithm is evaluated by comparing the performance of the algorithm with the "ideal" performance of the optimal policy using full state information. Extensive experiments are conducted, both numerically and in a network simulation tool that contains an implementation of the full wireless protocol stack. The results show that the Bayesian algorithm delivers close to optimal performance over a wide range of parameter values.

In Chapter 6 a CA network of parallel communication networks is considered where the foreground files are dynamically split among the networks in the presence of background traffic in each of the networks. Consequently, this approach is characterized as *dynamic job splitting*, because the splitting rule dynamically adapts based on full state information. In the corresponding job-split queueing model dynamic splitting is performed at infinitely fine-grained granularity in an effort to aim at "optimal" traffic splitting performance. Next, we present a practical realization of dynamic job splitting that uses a simple score function to make on-the-fly decisions on the routing of individual TCP segments. Then, we use a combination of the effective service time model from Chapter 2 and the PS-based dynamic job split model as a benchmark to evaluate the efficiency and practical usefulness of the practical realization of dynamic splitting TCP flows over real wireless networks in a test-lab environment. Extensive experimentation demonstrates that our solution is extremely efficient and easily deployable, and as such provides a powerful means to effectively split TCP traffic in the presence of concurrently available access networks.

In Chapter 7 the model from Chapter 2 is partially re-used for realizing a traffic control solution that aims at making QoS guarantees for wireless users in a network where the available capacity may fluctuate. It is shown that the throughput parameter bits/s does not provide sufficient insight in load conditions and/or traffic demands in wireless networks. To this end, we define a multi-service traffic profile that does provide this insight, which can be used to define the notion of a QoS budget. The QoS budget is assigned to a terminal in accordance with the method described in [47]. Using this method, the terminal can determine locally if the network consumption does not exceed the assigned budget and whether a new application session may fit within the QoS budget given. Finally, it is shown how the QoS budget with its multi-service traffic profiles can be used as a dynamic solution to guarantee the QoS of various ap-

plications in a wireless network, where channel conditions may vary over time and stations may move around.

In Chapter 8 a traffic policing solution is proposed and evaluated that is specifically developed for policing traffic of users in shared medium wireless networks. The solution described in Chapter 7 may be used to determine and assign a QoS budget to each terminal in a wireless network, based on a multi-service traffic profile. Subsequently, the QoS budget serves as an input to the traffic policing in Chapter 8. This policing solution takes advantage of the property of shared medium wireless networks that the upstream and downstream traffic compete for the medium when transmitting packets. This occurs in various types of practical wireless communication systems and leads to an inefficient medium utilization when a traffic contract maintains a strict separation between the two directions. In this chapter a method for traffic policing is introduced and evaluated that uses shared communication media more efficiently by exchanging the contract parameters between the uplink and the downlink directions dynamically, when desired. The solution is based upon a commonly used policing mechanism. An extensive performance study shows the merits of this policing mechanism and how it should be configured to use it effectively.

1.4 Publications

This thesis is based on the following publications that have appeared or have been submitted for publication in the open literature:

1. G.J. Hoekstra, R.D. van der Mei and S. Bhulai. Optimal job splitting in parallel processor sharing queues. To appear in Stochastic Models, 28(1), 2012.

2. G.J. Hoekstra, R.D. van der Mei and J.W. Bosman. On comparing the performance of dynamic multi-network optimizations. In Proceedings IEEE GlobeCom, Miami, U.S.A., 2010.

3. G.J. Hoekstra and R.D. van der Mei. Effective load for flow-level performance modelling of file transfers in wireless LANs. Computer Communications, 33(16):1972-1981, 2010.

4. F.J.M. Panken and G.J. Hoekstra. Multi-service traffic profiles to realise and maintain QoS guarantees in wireless LANs. Computer Communications, 32(6):1022-1033, 2009.

5. S. Bhulai, G.J. Hoekstra and R.D. van der Mei. Optimal concurrent access strategies in mobile communication networks. In Proceedings 22^{nd} International Teletraffic Congress, Amsterdam, The Netherlands, 2010.

6. S. Bhulai, G.J. Hoekstra, J.W. Bosman, and R.D. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A bayesian approach. Performance Evaluation, 69(1):4152, 2012.

7. G.J. Hoekstra, R.D. van der Mei, Y. Nazarathy and A.P. Zwart. Optimal file splitting for wireless networks with concurrent Access. Lecture Notes in Computer Science, 5894:189-203. Springer Verlag, 2009.

8. G.J. Hoekstra, R.D. van der Mei, T. Dziedzicki and J.W. Bosman. Efficient traffic splitting in parallel TCP-based networks: Modelling and experimental evaluation. Submitted for publication.

9. F.J.M. Panken and G.J. Hoekstra. On regulating traffic in shared wireless access media. Submitted for publication.

10. F.J.M. Panken, G.J. Hoekstra, D. Barankanira, J.C. Francis, R. Schwendener, O. Grondalen and M.G. Jaatun. Extending 3G/WiMAX networks and services through residential access capacity. IEEE Communications Magazine, 45(12):62-69, 2007.

11. G.J. Hoekstra and R.D. van der Mei. On the processor sharing of file transfers in wireless LANs. In Proceedings of the 69th IEEE Vehicular Technology Conference, VTC Spring 2009, Barcelona, Spain, 2009.

12. G.J. Hoekstra and F.J.M. Panken. On the efficient policing of HTTP traffic in WLANs. In Proceedings Third ERCIM Workshop on eMobility, pages 15-29, University of Twente, Enschede, The Netherlands, 2009.

13. G.J. Hoekstra and R.D. van der Mei. On the processor sharing properties of file transfers in a WLAN testbed. In Proceedings Internet-2010, pages 36-41, Valencia, Spain, 2010.

Chapter 2

Concept of Effective Service Time in Wireless Networks

Today, a wide range of wireless networks exist that provide users with data services. For engineering purposes there is a need for very simple, explicit, yet accurate, models that predict the performance under anticipated load conditions. In this context, several detailed packet-level models have been proposed, based on fixed-point equations. Despite the fact that these models generally lead to accurate performance predictions, they do not lead to simple explicit expressions for the performance. Instead, PS models may provide an abstraction from the highly complicated packet-level details of a network that reduces the model complexity significantly. By accounting for those essential factors that determine the network performance, highly accurate model predictions may be obtained, while an exact analysis of the model is still feasible.

Motivated by this, we propose a new concept of modeling the complex combined dynamics and protocol overhead of multiple layers into an explicit expression for a single parameter which will be called the *effective service time*. Based on the effective service time, we define the *effective load* to describe the flow-level performance of file transfers with an $M/G/1$-PS model. Despite the fact that PS models are heavily used in modeling flow-level performance in communication networks, an extensive validation of such models has not been published in the field. To this end, our model is validated extensively by comparing the model-based average response times against simulations and experiments conducted with network equipment.

The results show that the model leads to highly accurate predictions over a wide range of parameter combinations, including light- and heavy-tailed file-size distributions and light- and heavy-load scenarios. The simplicity and accuracy of the model make the results of high practical relevance and useful for performance engineering purposes. Another application of the proposed model is that it provides a mapping for parameterizing a PS-based queueing network, such that the outcomes predict the performance of real networks. Vice-versa the

model may be used to relate network performance measurements to outcomes obtained from queueing networks. It is exactly this property of providing a parameter mapping that makes our model particularly useful to relate the vast knowledge in the field of PS-based queueing networks to the area of network performance engineering. In Chapters 5 and 6 such parameter mappings are applied to different queueing networks to predict network performance.

This chapter is based on the results presented in [52], [51] and [50].

2.1 Introduction

PS models are frequently used to describe the bandwidth sharing of TCP flows in a network. A particularly attractive feature of PS models is that they abstract from the highly complicated packet-level details of the network, but at the same time maintain the essential factors that determine the performance, and also allow for an exact analysis. PS models have been successfully applied to model the flow-level behavior of a variety of communication networks, including CDMA 1xEV-DO [22], WLAN [76], UMTS-HSDPA [120] and ADSL [15].

An important property of PS models is the well-known insensitivity property of the mean response times with respect to the file-size distribution. In an excellent survey on statistical bandwidth sharing [105], it is stated that "Even though the conditions for the insensitivity properties of this model are not realized in practice, we can be fairly confident that *actual performance does not depend significantly on detailed flow and session characteristics*, given the assumption of Poisson session arrivals".

WLANs are widely deployed to provide users with wireless access to private or public data networks. Pioneering work on performance models for WLANs was done by Bianchi [21], who proposed a packet-level model for the saturated aggregated throughput of the Medium Access Control (MAC) layer. Combined packet/flow-level models have been proposed to study the performance of non-persistent data flows with [106, 87] or without [76] using the TCP protocol.

In [87], the authors have studied HTTP throughput performance in WLANs that operate in a specific (RTS/CTS) channel reservation mode with a detailed MAC model, combined with a state-dependent PS model. The proposed analytic model uses the fixed-point approach proposed by [21] and takes the TCP overhead associated with session set-up into account, specifically for HTTP applications. Users are assumed to alternate between activity periods (in which a page is downloaded) and idle periods. In these circumstances, the number of admitted HTTP sessions in the network is limited to the number of users. Another contribution [76] proposes an integrated packet/flow-level model for TCP flows in a WLAN, assuming that a station has no more than one active TCP flow at a time (similar to [108]).

In practice, most operational WLANs do not apply any form of per-flow admission control, operate in basic access mode (having disabled RTS/CTS), and allow stations to download multiple files concurrently. For performance engineering purposes there is a great need for simple, explicit, ready-to-implement yet accurate models that predict the file transfer performance over WLANs. However, previous work on flow-level performance [106, 108, 87, 76] meets at most partially these requirements. This has motivated us to develop a new, simple, yet accurate model that *does* take into account *each* of these requirements imposed by practical deployments. A first version of this model appeared in [50], followed by a detailed model in [51] that was thoroughly validated by OPNET simulations. In [52] the latter model is further extended and validated against testbed experiments.

Our main interest in this chapter is in the application-layer performance that depends on the combined dynamics of all underlying protocol layers. A crucial observation is that the combined dynamics of the MAC and TCP-layer yield a remarkably simple model, compared to separate and generally complex MAC [21] and TCP-layer models.

The main contributions of this chapter are as follows: First, we propose a new analytic flow-level model that translates the complex and detailed dynamics of the various protocol layers (i.e., FTP/TCP/IP/MAC), and their interactions, into an explicit expression for the 'effective service time' (denoted by β_{eff}) of the WLAN. Second, based on the effective service time we define the effective load (denoted ρ_{eff}) and use this to describe the flow-level behavior of TCP-based file transfers over WLANs (without admission control) as an $M/G/1$-PS model with load ρ_{eff}, instead of the classical load $\rho := \lambda\beta$, where λ is the arrival rate and β is the mean service time. Third, using the $M/G/1$-PS model we propose a simple analytic model to obtain the WLAN Access Point (AP) buffer-content distribution. A practically useful guideline for the minimum size of the AP buffer is given for the analytic flow-level model to apply. Fourth, we provide an extensive validation of the PS model by comparing the model-based outcomes against network simulations that implement the full range of lower-layer protocol details [91]. The simulation results demonstrate that the mean response times can be accurately predicted over a wide range of parameter combinations, including light- and heavy-tailed file-size distributions and light- and heavy-load scenarios. As a by-product, the simulation results demonstrate that the mean response times and AP buffer-content distribution are *indeed* fairly (but not completely) insensitive to the file-size distribution, as suggested by the $M/G/1$-PS model, which confirms the above-mentioned statement in [105].

The remainder of this chapter is organized as follows. In Section 2.2 we present a new analytic model for the flow-level behavior of file transfers for various types of WLANs, explicitly taking into account the details of the protocol stack at the MAC-layer and above. In Section 2.3 we validate the model via network simulations and testbed experiments, compare the outcomes against previous

work [106, 108] and our proposed analytic model. The outcomes from simulation and testbed experiments demonstrate near-insensitivity of the mean download response times to the file-size distribution. This section concludes with practical guidelines for performance engineering.

2.2 Modeling

WLAN performance models have received much attention from the research community of which the vast majority concentrated on the Distributed Coordination Function (DCF), as specified in the IEEE 802.11 standard [1]. The work by Bianchi concerned the initial version of the IEEE 802.11 standard for WLAN MAC and Physical (PHY) layer functionality. In 1999, the IEEE 802.11 standard was ratified [1] and served as the basis for higher data-rate amendments, known as the IEEE 802.11 b/a/g/n standards [3, 2, 5, 6]. As a result of using the same protocol basis, different parameterizations of Bianchi's model appeared [108, 35, 82] to capture the saturation throughput for the higher data-rate WLAN standards. The most prominent analytic models are based on the Markov chain approach of Bianchi [21]. Where Bianchi's parameterization of the model concerned the Frequency-Hopping Spread Spectrum (FHSS) method, others [108, 35, 82] have applied the PHY/MAC parameters of the IEEE 802.11 b-, g- and a-standard, respectively. It has been observed [76, 21] that the aggregate throughput performance strongly depends on whether medium access is provided in basic access or RTS/CTS mode. In the former case, the aggregate throughput strongly decays for an increasing number of active stations, whereas in the latter case the throughput performance is far less dependent on the number of active users.

In more recent work [93] a detailed description is given about the use of the IEEE 802.11 a/b/e/g/n standards and a mixture thereof in the context of multi-service QoS guarantees. For a detailed description of the PHY/MAC level aspects we refer to [93], of which the IEEE 802.11 MAC parameterization details are used as a basis for the analytic model presented below.

An important observation with regard to applying PS-based models to file downloads using TCP in infrastructure-based WLANs is made by Roijers et al. [106] and Sakurai and Hanley [108], who both state that the need to apply PS with state-dependent service rates (suggested by [22, 87] for the WLAN Medium Access Control (MAC)) *vanishes* when considering TCP flows in WLAN. This is because the stations hardly contend for the medium as the WLAN AP carries most of the traffic due to its equal medium access rights. This property allows us to assume that the total available capacity in the network is constant and to subsequently obtain the *effective service time* for a given average file size. A simple, explicit model for the effective service time is formulated in Section 2.2. This model lays the foundation for the notion of *effective load*, which captures the protocol dynamics in a single parameter that can be used to describe the

flow-level performance as a PS model.

Section 2.2.2 forms a brief introduction on modeling the IEEE 802.11 MAC aspects, followed by an analytic model suitable for network simulations in Section 2.2.3 that translates the main aspects from the MAC/IP/TCP-level dynamics into an explicit expression of effective service time at the application-layer. Further enhancements for the analytic model are proposed in Section 2.2.4 to describe the performance that can be observed in real network deployments. In Section 2.2.5 we use the latter expression to introduce the notion of effective load and use this to give an expression for the expected file transfer time for simulated and experimental networks.

2.2.1 Introduction to IEEE LAN standards

The 802.11 standard from the Institute of Electrical and Electronics Engineers (IEEE) is part of the 802 family of standards, which consists of several specifications that concern LAN technologies (see for example page 3 from [1], or page 389 from [57]). Each specification is indicated by a second number behind the 802 number. In 1980, Project 802 was started by the IEEE with the aim to define specifications for intercommunication of equipment from several manufacturers within the scope of the lower two layers of the OSI reference model; the data link layer and the physical (PHY) layer. The data link layer is subdivided into the Logical Link Control (LLC) and Medium Access Control (MAC) sublayers, of which the LLC sublayer is a common specification, indicated as IEEE 802.2. Every 802 network has a MAC-and a PHY (sub)layer, where the MAC defines the rules to gain access to the medium and to send data, and the PHY layer governs the transmission and reception of data to and from the MAC sublayer. In the remainder of this thesis, the MAC sublayer is for convenience referred to as a layer rather than sublayer although it is a sublayer in the context of 802 networks.

One of the most prominent examples of IEEE 802 standards specifications is the 802.3 specification that defines the Carrier Sense Multiple Access network with Collision Detection (CSMA/CD), which is commonly referred to as Ethernet. Similar to the IEEE 802.3 standard, the IEEE 802.11 standard is a link specification of the MAC-and PHY layer functionality and may be used for LLC/802.2 encapsulation. The IEEE 802.11 standard [1] defines the basic set of MAC-and PHY layer functionality that is enhanced in later standard amendments for i.e., higher transmission rates (IEEE 802.11 a/g/n), security (IEEE 802.11i) or Quality of Service enhancements (IEEE 802.11e). Compared to the wired IEEE 802 standards, the MAC-and PHY functionality is relatively complex due to the many additional features in the MAC and the use of the wireless medium at the physical layer. As a result, the PHY layer is split into two parts, first, the Physical Layer Convergence Protocol (PLCP) and, second, the Physical Medium Dependent (PMD) layer. The PLCP maps the MAC frames into an additional framing format suitable for the underlying PMD layer, and the

PMD defines the characteristics of, and method of transmitting and receiving data through the wireless medium. Any device that contains an IEEE 802.11 conformant medium access control (MAC) and physical layer (PHY) interface to the wireless medium is called a station.

In practice, a wide range of vendors deliver equipment that is conformant to one or more of the IEEE 802.11 standards on which contemporary WLANs are based. Certified compliancy to these standards means that the equipment of several vendors is tested to be interoperable. However, not all aspects that affect the performance of network equipment have been covered by the standard. For example, the standard does not specify the size of transmission buffers nor the method of dynamically adapting the transmission rate to changing channel conditions. Certain settings can be made to WLAN devices through user menus to change the functional behavior and the associated performance of the concerned device. When considering the performance of end-user applications, also the end-user equipment may have a decisive impact due to the hardware or software used or the settings applied to the various communication protocols.

In the remainder of this thesis the use of IEEE 802.11 systems is limited to equipment that operates as a wireless extension of a fixed or wireless infrastructure and therefore operates in, so called, infrastructure mode. Stations that provide access to the infrastructure are called APs and may allow other stations to associate to join the Basic Service Set (BSS), which consists of a set of stations and is ruled by a certain coordination function to distribute the access to the wireless medium and the infrastructure.

2.2.2 Introduction to IEEE 802.11 overhead modeling

The CSMA/CA scheme to access the medium is used by IEEE 802.11-compliant stations (abbreviated as STA) that use the DCF. In Figure 2.1 a transmission cycle is shown in which two WLAN STAs (an AP and its associated station) exchange several MAC data frames, Data1-3. These MAC data frames are formally indicated as MAC Protocol Data Units (MPDUs) [1]. When a STA wants to transmit an IP packet, it is encapsulated in a MAC data frame and the STA will sense if the medium is busy or not. If the medium is idle and remains so during the following Distributed InterFrame Space (DIFS) period, the transmission may proceed. If the medium is determined to be busy, the STA waits until the end of the current transmission. If the medium is not used for a subsequent DIFS period the STA generates a random backoff period, unless the backoff timer already contains a nonzero value. Subsequently, the STA decrements its backoff timer for every time slot, τ, waited during this backoff period (indicated by $Cw1-3$). If the backoff timer reaches zero, transmission may commence.

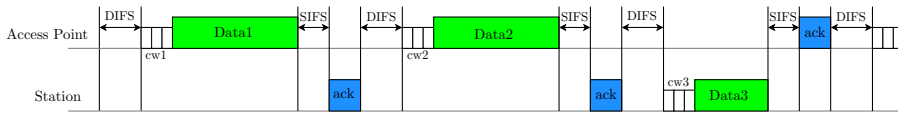


Figure 2.1: Transmission cycle of an IEEE 802.11-compliant station.

The random backoff time is generated from a uniform distribution, $[0, ..., Cw]$. Here the contention window parameter, Cw, is an integer within the range Cw_{min} and Cw_{max} that are defined by the various PHY standards of WLAN (cf. [3, 2, 5]). Initially, Cw is set to Cw_{min}, on every unsuccessful transmission attempt the next Cw values take the sequentially ascending integer powers of 2, minus 1, up to and including the Cw_{max} value. On every successful transmission attempt Cw is reset to Cw_{min}. On correctly receiving a MAC data frame the destination STA will send a MAC acknowledgment (ACK) control frame to the source STA after waiting for a Short InterFrame Space (SIFS) period. Subsequently, new data transmissions may follow after sensing the medium idle again as is illustrated in Figure 2.1. Note that the data frame is a unicast transmission, that must be acknowledged with a MAC ACK frame.

If a source STA does not receive an ACK within the ACK timeout (as defined in [1]), the source station must perform a retransmission of its last MAC data unit. Retransmissions may be repeated up to the maximum retransmission threshold and cause the contention window to increase up to the value of Cw_{max} is reached. On receiving an erroneous MAC data unit (e.g. due to a collision), STAs backoff a sufficient amount of time equal to the Extended InterFrame Space (EIFS). The purpose of this backoff is to reserve enough time for a MAC ACK transmission on what was the incorrectly received frame. If a source does not receive a MAC ACK, the packet is assumed to be lost and will be retransmitted after the EIFS period. This packet retransmission follows the same procedure as on the expiration of the ACK timeout after a correct transmission. In addition to the data-and control frames, a third type of frame may be transmitted by the AP for channel management purposes. These management frames are formally indicated as Management MPDUs (MMPDU) [1], some of which are broadcasted on the medium and are therefore not acknowledged by the receiving stations.

Certain IEEE 802.11 PHYs offer the possibility for multi-rate transmissions and allow dynamic rate switching to improve the network performance. The switching algorithm itself is beyond the scope of the IEEE 802.11 standard. For the sake of interoperability certain rules must be followed by stations for transmitting the different types of frames at the various rates. Data frames directed to another destination station may be transmitted at any of the rates supported by the receiver, whereas control frames must be transmitted at a rate that is

supported by all stations. In general, management frames may be transmitted at a broad set of rates, similar to data frames. However, the purpose of a management frame may vary and so may their transmission rate.

2.2.3 Analytic model for network simulations

We assume a network consisting of several WLAN stations and one AP, in which the stations are downloading files from an FTP server that is located close (with small propagation delay) to the AP. Each station generates FTP download requests according to a Poisson process and may have multiple file transfers in progress because there is no admission control mechanism on the number of file transfers per station or in total. All file transfers are carried over TCP connections that use delayed acknowledgments.

Our model accounts for all overhead associated with a file download; the file transfer itself, the FTP commands and TCP handshake for opening and closing sessions on a WLAN network operating in basic access mode, using the DCF. As explained in the previous section, STAs operating in CSMA/CA-mode that sense the medium busy must first decrement their backoff timer prior to initiating their packet transmission. Similar to [106], it is assumed in our model that a STA must perform a backoff before transmitting a TCP ACK because it finds the medium busy when transmitting its MAC ACK on the previously received TCP data segment to be acknowledged by TCP. Furthermore, the impact of collisions on the length of the backoff is not taken into account because the probability is small and the collided stations will perform their retransmissions in competition again, and thus leaving the medium idle until the first station may proceed. Finally, we assume that the handling of download requests imposes such limited CPU requirements that it can be ignored in comparison to the delay imposed by the wireless network.

Practice shows that these assumptions are reasonable. Packet loss at the IP-layer is not accounted for by the model because (1) the WLAN protocol retransmits lost packets, and (2) the medium contention is low in the circumstances we consider, see the remainder of this section. The observations and model fundamentals from [106] are used as a basis for the model proposed in this chapter, but with the following extensions and modifications:

1. The flow-level model assumes the absence of admission control.
2. The model is generalized towards contemporary higher rate supplements (IEEE 802.11 a/g/n).
3. The protocol overhead is modeled in more detail.
4. The backoff period modeling is refined.
5. Several MAC-level parameters are corrected.

One may suspect that the above-mentioned modifications to the model in [106] lead to a better estimation of the file download times (see Section 2.3 for a validation).

When considering the IEEE 802.11a/g/n standards, one can express the time, $T_d(x)$, spent by a WLAN station on transmitting a TCP *data* segment of x bits and its associated IEEE 802.11 MPDU, IP and TCP overhead as:

$$T_d(x) = PHY + \left\lceil \frac{MAC + X_{hloh} + x}{TS \cdot STT} \right\rceil \cdot STT, \tag{2.1}$$

with MAC the IEEE 802.11 MAC-layer overhead bits, SST the Symbol Transmission Time and TS the transmission rate or speed used by the sending station. The overhead associated with the higher-layer protocols of the IEEE 802.11 MAC is represented by X_{hloh}. In the case of using TCP and IP as higher-layer protocols X_{hloh} is set to $X_{tcp/ip}$ bits. Note that the TCP segment and the TCP/IP/MAC overhead are transmitted in a whole number of transmission symbols. Additionally, a fixed amount of time, PHY, is consumed on the medium by transmitting the overhead associated with the Physical Layer Convergence Protocol (PLCP) and by the signal extension (depending on the standard used).
We refer to Table 2.1 for representative parameters of the various IEEE 802.11 standards. Note that these sets may differ from one network to another because the IEEE 802.11 standards allow optional capabilities and different system configurations.

After successful transmission of a data segment, the destination WLAN station is considered to reply by transmitting a MAC *acknowledgment* control frame. This occupies the medium for T_c seconds:

$$T_c = PHY + \left\lceil \frac{ack}{TS_c \cdot STT} \right\rceil \cdot STT, \tag{2.2}$$

where instead of the MAC overhead bits a smaller ack overhead and a transmission rate, TS_c, is used. Note that TS_c depends on the transmission rate of the data frame to be acknowledged and on the BSS basic rate set configuration of the network, which defines a pre-configured set of transmission rates that must be supported by all stations in the BSS. Management frames of size X_m may be transmitted on the medium at a transmission rate equal to TS_m (bps) that depends on the BSS basic rate set configuration. The transmission of a management frame of size x occupies the medium for $T_m(x)$ seconds:

$$T_m(x) = PHY + \left\lceil \frac{X_m}{TS_m \cdot STT} \right\rceil \cdot STT. \tag{2.3}$$

For the IEEE 802.11b standard, the following, very similar, equations apply to the time spent on transmitting the same TCP *data* segment, $T_d(x)$, on the

associated WLAN acknowledgment, T_c, and on a WLAN management frame, $T_m(x)$:

$$T_d(x) = PHY + \frac{MAC + X_{hloh} + x}{TS}, \tag{2.4}$$

$$T_c = PHY + \frac{ack}{TS_c}, \tag{2.5}$$

$$T_m(x) = PHY + \frac{x}{TS_m}, \tag{2.6}$$

where TS represents the WLAN transmission rate (in bps) for data segments and TS_c for the WLAN acknowledgments and TS_m management frames. When WLAN stations operate in basic access mode, source and destination stations should wait for certain inter-frame spacing times ($DIFS$ and $SIFS$) between the transmission of WLAN MAC data and acknowledgment frames. Time $T_{da}(x)$ is defined as the time needed for MAC-acknowledged reception of a TCP segment consisting of x bits, taking into account a propagation delay of T_p seconds:

$$T_{da}(x) = DIFS + T_d(x) + T_p + SIFS + T_c + T_p. \tag{2.7}$$

Depending on the IEEE 802.11 standard, $T_d(x)$ and T_c, may either be used from (2.1) and (2.2) or alternatively from (2.4) and (2.5). When applying TCP with delayed acknowledgments for acknowledging (the de facto) every other TCP data segment, the data is transmitted by repeated execution of a transmission cycle, encompassing the transmission of two TCP data segments by the source station and one TCP ACK segment by the destination station (depicted in Figure 2.1). During such a transmission cycle typically one station and the AP contend for the medium to send a TCP ACK and two TCP data segments respectively. Consequently, the AP is responsible for transmitting approximately $^2/_3$ of all packets in the network.

As explained in [106, 108] the collision probability (and the total TCP throughput) is insensitive to the number of ongoing file transfers: as all flows pass through the AP, while the AP has equal MAC rights and develops a large backlog of packets in its transmission buffer. When the WLAN MAC behavior is combined with the TCP acknowledgment mechanism, only the station that has received two TCP data segments from the AP is enabled by TCP to contend for the medium. TCP will simply inhibit all other stations (apart from those initiating a new transfer) to become active on the WLAN MAC. As a result, packet loss at the IP-layer can be neglected in our analysis because the WLAN protocol will retransmit lost packets.

In the process of contending for the medium, WLAN stations must wait a backoff time before initiating their data transmission. Still collisions may be experienced. However, in view of the observation by [106] that there is, in addition to the AP, only one station contending for the medium, we may expect

that the backoff distribution is bounded by Cw_{min}. Some authors approximate the expected time consumed by the two backoff periods within a cycle by $\frac{Cw_{min}}{2}$ time slots. However, [108] specifies a more accurate analysis reasoning that for non-delayed TCP ACKs, the average backoff contribution in the file transfer transmission cycle will be the maximum of two independent observations from the uniform minimum backoff distribution $[0, ..., Cw_{min}]$.

For TCP with delayed acknowledgments this means that two of the three backoff periods will contribute on average with $\frac{Cw_{min}(4Cw_{min}+5)}{6(Cw_{min}+1)}$ time slots, where it should be noted that Cw_{min} is defined, in accordance with the standard [3] and in contrast to [106], as the maximum value of the minimum backoff interval and equals 31 slots for an IEEE 802.11b PHY.

During the remaining period the AP is the only active station and will backoff, on average, $\frac{Cw_{min}}{2}$ slots after each successful transmission. As a result it can be expected that there is one backoff period per cycle in which both stations collide with probability $\frac{1}{Cw_{min}+1}$. This allows formulating the total expected time of a transmission cycle of two TCP data segments and one TCP ACK segment as:

$$T_{cycle} = 2T_{da}(X_{MSS}) + T_{da}(0) + \frac{Cw_{min}\,(7Cw_{min}+8)\,\tau}{6(Cw_{min}+1)} + \frac{T_{col}}{Cw_{min}+1}, \tag{2.8}$$

$$T_{col} = T_d(X_{MSS}) + T_p + EIFS, \tag{2.9}$$

where T_{cycle} is the expected time of an entire transmission cycle during the file transfer, and T_{col} is the time involved in a collision on the medium. The remaining parameters are Cw_{min} (minimum contention window), τ (slot time), X_{MSS} (TCP Maximum Segment Size (MSS)). Note hat the TCP ACK segment does not transport any TCP data bits and thus requires an expected time of $T_{da}(0)$ for MAC-acknowledged reception.

Parameter	802.11a	802.11b	802.11g	802.11n
Cw_{min}(slots)	15	31	15	15
MAC(bits)	246	224	246	246
τ	9μs	20μs	9μs	9μs
SIFS	16μs	10μs	10μs	16μs
DIFS	34μs	50μs	28μs	34μs
EIFS	90μs	364μs	342μs	90μs
PHY	20μs	192μs	26μs	20μs
ack(bits)	134	112	134	134
T_p	1μs	1μs	1μs	1μs
TS(bps)	$54 \cdot 10^6$	$11 \cdot 10^6$	$54 \cdot 10^6$	$54 \cdot 10^6$
TS_c, TS_m(bps)	$24 \cdot 10^6$	10^6	$24 \cdot 10^6$	$24 \cdot 10^6$
STT	4μs	NA	4μs	4μs

Table 2.1: Summary of important parameters in IEEE 802.11 protocols.

When considering the file download response time, a certain amount of time is consumed by the file transfer. The remaining part of the traffic is exchanged for initiating and closing a TCP connection and for issuing the FTP commands and has much greater impact if file transfers become shorter and are hardly considered in WLAN flow-level performance models. TCP connection initiation involves a 3-way handshake of TCP (SYN) segments and for closing the sessions a 4-way handshake (FIN, ACK) is used [110]. In the interest of simplicity, the FTP application is modeled to use one TCP session for the FTP commands and the file transfer.

During the TCP set-up cycle, a station starts by initiating a TCP SYN segment, which is followed by a SYN ACK segment by the AP and is concluded by a TCP ACK from the station. The AP will acknowledge on the WLAN medium the packet carrying the TCP SYN from the station. As the AP has always packets to transmit, the SYN ACK is highly likely to be transmitted after and before a pair of TCP data segments from other flows.

When a station is transmitting the first (TCP SYN) and third (TCP ACK) segment a collision may occur with a small or large packet, assumed with equal probability. The expected time of a collision during these two sequences are $(T_{shortcol} + T_{col})/2$. The second (SYN ACK) segment may collide only with a TCP ACK and thus contributes $T_{shortcol}$ to the expected collision time during TCP set-up, which explains the last term in (2.10). The total expected time spent in backoff by the AP after transmitting its SYN ACK is assumed equal to the minimum of the backoff windows drawn by the AP and a station, corresponding to the expectation of the minimum of two uniformly i.i.d. observations from the backoff interval, thus contributing with an expected delay of $\frac{Cw_{min}(2Cw_{min}+1)}{6(Cw_{min}+1)}$ time slots. The total time spent on average in a TCP set-up

then equals:

$$T_{tcp\_setup} = 3T_{da}(0) + \frac{Cw_{min}(2Cw_{min}+1)\tau}{6(Cw_{min}+1)} + \frac{(2T_{shortcol}+T_{col})}{Cw_{min}+1}, \tag{2.10}$$

$$T_{shortcol} = T_d(0) + T_p + EIFS. \tag{2.11}$$

As soon as the TCP connection is established, the station issues the FTP GET command. It is assumed that the FTP GET command (assumed equally sized as trivial ftp (tftp) requests) has a size of 512 bytes or 4096 bits(X_{FTP}). Since the station's backoff time does not inhibit the AP from using the medium it is not explicitly modeled, the average time spent on using the medium for the FTP GET request equals:

$$T_{FTPget} = T_{da}(X_{FTPget}) + \frac{T_{col}}{Cw_{min}+1}. \tag{2.12}$$

Note that the TCP ACK on the FTP GET request is not modeled here because the first TCP data segment of the file transfer will piggyback the TCP ACK and is already accounted for in the file transfer transmission cycle. As both the AP and the station compete for the medium, the collision probability is included in the above expression. The file transfer is concluded by the transmission of the last data segment, which is immediately followed by an FTP close command with an assumed size of 8 bytes ($X_{FTPclose}$). The expected size of the last data segment of the file (for non-deterministic file-size distributions) approximately equals $\frac{X_{MSS}}{2}$, and hence:

$$T_{lastcycle} = T_{da}(X_{FTPclose}) + T_{da}\Big(\frac{X_{MSS}}{2}\Big) + T_{da}(0) + \frac{T_{halfMSSCol}}{Cw_{min}+1} + \frac{Cw_{min}\,(7Cw_{min}+8)\,\tau}{6(Cw_{min}+1)} + \frac{1}{2}\Big(T_{da}(0) + T_{shortCol} + \frac{Cw_{min}(2Cw_{min}+1)\tau}{6(Cw_{min}+1)}\Big), \tag{2.13}$$

$$T_{halfMSSCol} = T_d\Big(\frac{X_{MSS}}{2}\Big) + T_p + EIFS. \tag{2.14}$$

After sending the last TCP data segment (FTP close command), the AP will contend with the station that attempts to transmit its last TCP ACK and later sending its TCP FIN. As an even number is considered equally likely as an odd number of data segments per transfer, the additional overhead related to sending an additional TCP ACK is accounted for this proportion accordingly. As a result of the connection closure, two cycles need to follow in which the station and the AP contend for the medium and concurrently decrement their backoff timer. Possible collisions ($T_{shortcol}$) will be shorter as the most likely involved segments contain no data bits. The expected time to close the TCP

connection can then be expressed as:

$$T_{TCP\_close} = 4T_{da}(0) + \frac{2Cw_{min}(4Cw_{min}+5)\tau}{6(Cw_{min}+1)} + \frac{2T_{shortcol}}{Cw_{min}+1}. \tag{2.15}$$

Note that the TCP closure was not accounted for in the analysis of [87] because the closure of the TCP connection does not affect the download response time, however, it needs to be noted that the TCP set-up, closure and FTP commands do contribute to the contention and overall load on the network and to a lesser degree to an overestimation of individual download response times. The expected time consumed by the FTP commands and the TCP session opening/closing as defined by (2.10), (2.12) and (2.15), can be expressed as:

$$T_{tcpftp_{OH}} = T_{tcp\_setup} + T_{FTP\_get} + T_{TCP\_close}. \tag{2.16}$$

Now, we obtain the effective service time, β_{eff}, of the file transfer as observed at the application-layer by combining (2.8), (2.13), and (2.16):

$$\beta_{eff} = \frac{\left(X_{file} - \frac{X_{MSS}}{2}\right)T_{cycle}}{2X_{MSS}} + T_{lastcycle} + T_{tcpftp_{OH}}, \tag{2.17}$$

with X_{file} as the mean file size (in bits). Note that in our modeling approach, we have assumed the use of TCP with delayed acknowledgments. Although this meets many practical circumstances, some TCP implementations acknowledge each individual data segment (instead of every two segments). To account for this effect, our model may be adapted to the use of non-delayed acknowledgments by replacing T_{cycle}, $T_{lastcycle}$ and β_{eff} from (2.8),(2.13) and (2.17) by $\hat{T}_{cycle}$, $\hat{T}_{lastcycle}$ and $\hat{\beta}_{eff}$, respectively, defined as follows:

$$\hat{T}_{cycle} = T_{da}(X_{MSS}) + T_{da}(0) + \frac{Cw_{min}\,(4Cw_{min}+5)\,\tau}{6(Cw_{min}+1)} + \frac{T_{col}}{Cw_{min}+1}, \tag{2.18}$$

$$\hat{T}_{lastcycle} = T_{da}(X_{FTPclose}) + T_{da}\Big(\frac{X_{MSS}}{2}\Big) + 2T_{da}(0) + Cw_{min}\tau + \frac{T_{halfMSSCol} + T_{shortCol}}{Cw_{min}+1}, \tag{2.19}$$

$$\hat{\beta}_{eff} = \frac{\left(X_{file} - \frac{X_{MSS}}{2}\right)\hat{T}_{cycle}}{X_{MSS}} + \hat{T}_{lastcycle} + T_{tcpftp_{OH}}. \tag{2.20}$$

Now the transmission cycle in (2.18) comprises the transmission of one TCP data segment followed by one TCP acknowledgment. Consequently, the average backoff contribution now corresponds to the two periods (identified in

the paragraph preceding (2.8)) and does not affect the expression for the collisions because the AP was not competing with the station for the second packet transmission that is now eliminated from the transmission cycle. For the last transmission cycle, the last data segment (with expected size $\frac{X_{MSS}}{2}$) is now always acknowledged separately from the FTP close command. The resulting expected backoff contribution is obtained by combining the one from (2.18) and the backoff associated with the TCP ACK transmission from (2.19) that was used for an even number of data segments.

2.2.4 Analytic model enhancements for testbed experiments

In the previous section, the main aspects for modeling file transfers over a WLAN network have been accounted for. In practice, however, the WLAN network also carries management traffic and applies frame encapsulation that is not commonly accounted for due to the limited impact this is considered to have on the overall network performance [45]. Moreover, the FTP application is in reality more sophisticated than described in the previous section.

Although the model described previously closely matches the behavior of real networks for most purposes and for the simulation studies conducted in subsequent chapters, further model enhancements are proposed in this section to improve the analytic model specifically for testbed experiments that expose the network to high traffic loads. Exactly in these circumstances certain aspects that may seem of minor importance turn out to have a noticeable performance impact. In the following subsections the model enhancements for use in practical deployments are presented.

Frame encapsulation overhead

The Logical Link Control (LLC) layer as defined by the IEEE 802.2 standard can be used by underlying protocol (sub)layers, therefore the use of LLC is optional; some implementations of the IEEE 802.11 standard do not use the LLC layer (such as the OPNET modeler implementation), whereas other implementations (i.e., from equipment manufacturers like Cisco and Linksys) do use the LLC layer. In the latter case, two encapsulation methods may be applied, one method described in RFC 1042 [101] and the other in the IEEE 802.1H standard [7], both are derivatives of the IEEE 802.2 Subnetwork Access Protocol (SNAP) and introduce eight bytes of additional header information to the WLAN MAC payload.

In the context of the presented performance model, there are no implications of the LLC/SNAP encapsulation as a higher-layer protocol other than a reduced medium efficiency. Since IEEE 802.11-based WLANs are capable of transmitting MAC frames with payloads up to 2304 bytes per frame, IP packets may still have a size of 1500 bytes, in accordance with RFC 1191 that defines the Path Maximum Transmission Unit (MTU) discovery for IP networks. Therefore, the

overhead due to the LLC/SNAP encapsulation adds to the other encapsulation overhead and can be accounted for by using $X_{hloh} = X_{tcp/ip} + 64$ in (2.1) and (2.4).

FTP modeling

The current File Transfer Protocol (FTP) is standardized since 1985 in RFC 959 [100]. The first file transfer mechanisms were already proposed in 1971 and implemented on hosts at MIT, followed by many RFCs and other implementations. As opposed to the implementation in the OPNET network simulation environment, practical deployments operate in accordance with the methods outlined in RFC 959 as is done in the widely used ProFTPD [102]. To obtain more accuracy in modeling experimental deployments, more detailed FTP interactions are added to the analytic model that are outlined below.

As we assume that all users have logged in to the FTP server already, the message sequences needed to setup a file download involve (cf. [110]) the transmission of an FTP passive command (PASV) with size $X_{ftp-pasv}$ by the station, followed by a 227 (entering passive mode) response from the server (with a piggy-backed TCP ACK) of size $X_{ftp-227}$ by the FTP server (and hence the AP) to confirm the passive file download mode. This two-way handshake is modeled similar to the first two sequences of the TCP setup in (2.10), but with the appropriate TCP segment payload values for the PASV request and the 227 response.

$$T_{ftp\_setup} = T_{da}(X_{ftp-pasv}) + T_{da}(X_{ftp-227}) + \frac{Cw_{min}(2Cw_{min}+1)\tau}{6(Cw_{min}+1)} + \frac{(T_{shortcol}+T_{col})}{Cw_{min}+1}. \tag{2.21}$$

Here, the idle time of the medium due to backoff is expected to be the minimum of two backoff observations because the AP and the station have both a packet to transmit. The PASV request may collide with a TCP data segment from the AP with the probability of both stations drawing the same backoff interval, whereas the 227 response by the AP may only collide with the smaller packets from the stations. After receiving the 227 (entering passive mode) response from the FTP server, the station initiates a new TCP connection for the data transfer and issues an FTP retrieve command (RETR) with size $X_{ftp-retr}$ to obtain a certain file.

As soon as the TCP connection is established, the station issues the FTP RETR command and will receive a 150 (opening binary mode data connection) response message of size $X_{ftp-150}$ from the FTP server. Since the station's backoff time does not inhibit the AP from using the medium, the station's backoff time is in this case not explicitly modeled. The average time spent on using the medium

for the FTP RETR Request and Response (RR) equals:

$$T_{ftp\_rr} = T_{da}(X_{ftp-retr}) + T_{da}(X_{ftp-150}) + \frac{Cw_{min}(2Cw_{min}+1)\tau}{6(Cw_{min}+1)} + \frac{T_{col}+3T_{shortcol}}{2(Cw_{min}+1)}. \quad (2.22)$$

Directly after the 150 response (opening binary mode data connection) is transmitted by the FTP server, the TCP transmission cycle is repeated up to the point where the last cycle is transmitted. Unlike the model in Section 2.2.3 we assume in the present model that the user's FTP session remains active and does not require the transmission of an FTP closure command. Hence the last transmission cycle may consist of one or two TCP data segments, with equal probability, and the size of the last TCP data segment will on average be $X_{MSS}/2$ bits. We therefore conclude that the last TCP transmission cycle transports on average X_{MSS} bits in an expected time equal to:

$$\tilde{T}_{lastcycle} = \frac{T_{da}(X_{MSS})}{2} + T_{da}\left(\frac{X_{MSS}}{2}\right) + T_{da}(0) + \frac{T_{halfMSSCol}}{Cw_{min}+1} + \frac{Cw_{min}(11Cw_{min}+13)\tau}{12(Cw_{min}+1)}. \quad (2.23)$$

In (2.23) the last term of $\tilde{T}_{lastcycle}$ represents the idle time of the medium due to backoff and accounts for the fact that the last cycle may consist of one or two TCP segments. As a result, two or three backoff periods may occur during the last cycle with equal probability.

After the station has acknowledged the last TCP data segment, it will close the TCP data connection by transmitting a TCP FIN that is acknowledged by the FTP server, followed by an FTP 226 (transfer complete) response of size $X_{ftp-226}$ to indicate that the transfer has completed. Finally, the station acknowledges this response, which concludes the data transfer. Similar to the TCP connection setup (2.10) and the FTP retrieve request (2.22) the AP contends with at least one station for the medium, causing the medium idle time to be equal to the minimum of two backoff windows and possible collisions occur with small packets. Accordingly, the station's backoff does not inhibit the medium from being used and the transmissions may collide with the larger TCP data segments from the AP. The expected time to close the FTP data transfer can be approximated by:

$$\tilde{T}_{TCP\_close} = 3T_{da}(0) + T_{da}(X_{ftp-226}) + \frac{2Cw_{min}(2Cw_{min}+1)\tau}{6(Cw_{min}+1)} + \frac{2T_{shortcol}+2T_{col}}{Cw_{min}+1}. \quad (2.24)$$

The time consumed by transmitting the FTP overhead and TCP session opening/closing as defined by (2.10), (2.21), (2.22) and (2.24) is calculated by:

$$\tilde{T}_{tcpftp_{OH}} = T_{ftp\_setup} + T_{tcp\_setup} + T_{ftp\_rr} + \tilde{T}_{TCP\_close}. \quad (2.25)$$

Now, we obtain the effective service time, $\tilde{\beta}_{eff}$, of the file transfer as observed at the application-layer for the enhanced analytic model by combining (2.8), (2.23), and (2.25):

$$\tilde{\beta}_{eff} = \frac{\left(X_{file} - \frac{X_{MSS}}{2}\right)T_{cycle}}{2X_{MSS}} + \tilde{T}_{lastcycle} + \tilde{T}_{tcpftp_{OH}}, \tag{2.26}$$

with X_{file} as the mean file size (in bits). Note that in our modeling approach, we have assumed the use of TCP with delayed acknowledgments.

WLAN beacon frames

Many performance models have appeared in the literature that are validated by simulations (cf. [76, 106, 87, 108]) and have concentrated on the main aspects of the medium access control protocol. In [45] the authors observed that other performance models ignored the impact of among others the Timing Synchronization Function (TSF). In an infrastructure network, the APs are responsible for performing the power management function by distributing the time in certain management frames, called beacons. When APs are about to transmit a beacon frame, their local time is inserted into the beacon. The IEEE 802.11 standard mandates that the AP periodically transmits the beacons and that the receiving stations always accept the timing information received.

Another purpose of the beacon frame is to announce the presence of an IEEE 802.11 network and its capabilities at regular intervals, called Beacon Intervals (BI). These time intervals are expressed in Time Units (TU) of 1.024 microseconds. Typically, beacon intervals are set to 100 TUs. The beacon frame is based on an IEEE 802.11 management frame, which uses the same amount of MAC overhead bits than for the regular data frames. Similar to the transmission of data frames, the AP follows the procedure described in Section 2.2.2 for the transmission of beacon frames. However, the receiving stations do not acknowledge the reception of the beacons that are broadcasted by the AP. To this end, the transmission cycle of a beacon frame, T_{bc}, is approximated by using (2.3) or (2.6) as follows:

$$\tilde{T}_{bc} = DIFS + T_m(X_{beacon}) + T_p + \frac{Cw_{min}\tau}{2},$$

where X_{beacon} is the size of the entire management frame including the beacon specific fields, as defined at page 45 of [1]. Although the transmission of beacons may be delayed due to CSMA deferrals, subsequent beacons have to be scheduled by the AP at the nominal interval, even on a busy network. Consequently, the effective service time of the file transfer as observed at the application-layer is stretched. The effective service time in the absence of beacon frames denoted β'_{eff} that may be obtained from either (2.17), (2.20), (2.26) is treated as follows

to obtain the effective service time, β''_{eff}, that accounts for the presence of beacon frames:

$$\beta''_{eff} = \beta'_{eff}\Big(1 + \frac{T_{bc}}{BI}\Big). \tag{2.27}$$

2.2.5 Effective load

To model the flow-level behavior of file transfers, we consider a classical $M/G/1$-PS model, with flow-arrival rate λ, and where the service time B is generally distributed with mean β. In this model, incoming flows immediately enter the system, thereby receiving a fair share of the available capacity. Then the occupation rate is $\rho := \lambda\beta$, and the expected sojourn time is known to be $\mathbb{E}[S] = \beta/(1-\rho)$. Note here that the sojourn time is *insensitive* to the service-time distribution, i.e., depends on the service-time distribution only through its mean β.

To translate the analytic model for WLAN file downloads (discussed above) into an $M/G/1$-PS model, we define the following notion of *effective load*:

$$\rho_{eff} := \lambda \cdot \beta_{eff}, \tag{2.28}$$

where β_{eff} is the 'effective service time' defined in (2.17) for the analytic model suited for network simulations that uses TCP with delayed acknowledgments. Similarly, the effective load is obtained for TCP implementations that use non-delayed acknowledgments or are based on a model enhancement suited for experimental purposes by using the effective service time definitions from (2.20) and (2.26) and (2.27). The quantity ρ_{eff} can be viewed as the *effective medium utilization* resulting from the load introduced by processing λ file download requests per second. Since the file download in the WLAN network encompasses the file transfer and the introduced overhead of FTP and TCP, the expected file transfer time (denoted $\mathbb{E}[R]$) is modeled as the expected sojourn time in an $M/G/1$-PS model with load ρ_{eff}:

$$\mathbb{E}[R] = \frac{\beta_{eff}}{1-\rho_{eff}}. \tag{2.29}$$

Thus, to apply the analytic model, the average file download time $\mathbb{E}[R]$ is obtained from (2.29), where ρ_{eff} is given by (2.28), and β_{eff} follows from (2.17). Note that β_{eff} in (2.28) and (2.29) should be replaced by $\hat{\beta}_{eff}$ from (2.20) when using non-delayed TCP acknowledgments instead of delayed acknowledgments. To account for the analytic model enhancements from Section 2.2.4, β_{eff} in (2.28) and (2.29) should be replaced by (2.26).

In [67] it is shown that the steady-state distribution of N, the number of jobs in an $M/G/1$-PS system (defined earlier), is given by, for $n = 0, 1, \ldots$,

$$P\left(N = n\right) = \left(1-\rho\right)\rho^n, \tag{2.30}$$

independent of the service-time distribution (for given β). In the context of file transfers in WLAN, we view the number of customers in an $M/G/1$-PS system as the number of file downloads in progress. For relating the number of file downloads to the buffer content distribution, the TCP bandwidth-delay product is applied as follows:

$$w \geq C \cdot RTT, \tag{2.31}$$

where C represents the available bandwidth of a TCP connection (in bps), RTT the round-trip time (in seconds), w the TCP maximum window size (in bits). For considering a TCP connection with a continuous flow of data without waiting times, w should meet the above requirement [104]. This requirement can be easily met under the conditions considered in this chapter: based on the TCP configurations used and an overly optimistic C of 5 Mbps, the RTT may exceed 10 ms, which supports the applicability of (2.31) in this context. Let the random variable X denote the number of TCP data segments in the WLAN AP transmission buffer. Then the number of TCP segments in the buffer can be approximated by the following relation:

$$X \approx N \cdot \lfloor w/X_{MSS} \rfloor, \tag{2.32}$$

where N is the number of file downloads in progress and X_{MSS} the TCP MSS. Then, in the spirit of (2.30), the probability distribution of X is approximated by: for $n = 0, 1, \ldots$,

$$P\left(X = n \cdot \lfloor w/X_{MSS} \rfloor\right) \approx (1 - \rho_{eff})\, \rho_{eff}^{n}, \tag{2.33}$$

using the assumption that the TCP connections carrying the file download traffic have their maximum window size, w, of data segments in flight. Most of these data segments reside in the AP transmission buffer; one may be in transfer between the two stations and perhaps another segment may already have been received and should be acknowledged for when receiving the next (when using TCP delayed acknowledgments). Note that the dynamics of the TCP segments in the AP transmission buffer and the number of downloads in the network operate on different time scales.

2.3 Model validation by simulation

2.3.1 Experimental setup

As WLAN networks all rely on the same MAC protocol basis, the MAC parameters of our model are based on the IEEE 802.11b standard amendment for its wide availability and lower computational requirements involved in high-load simulation validation. In contrast to previous contributions [108, 106, 87] we validate our model with OPNET Modeler (v14.5) [91], rather than using ns-2 simulator. OPNET contains a standard library of detailed WLAN models, including an AP, that may also serve as application server, and wireless stations

that are used for downloading files from the AP. Our simulated network consists of 11 nodes; an AP is surrounded by 10 stations spaced at an equal distance of 15 meters. All nodes are configured to use the IEEE 802.11b 11 Mbps transmission rate. FTP requests arriving from the stations at the AP are handled by the application-layer of the AP. As these simulation models take many more parameters into consideration than our proposed analytic model, the simulation scenarios require careful configuration.

To this end, the impact of the central processing unit (CPU) on file transfers is eliminated by assuming infinitely fast CPUs in all devices. Our model is validated for the widely used Reno TCP configuration [112] and an enhanced version of Reno indicated in OPNET as *Full-Featured*. In general, the TCP stack implements the basic standard of TCP from RCF-793 [99] and RFC-1122 [27] on the requirements on Internet hosts and uses slow start, congestion avoidance, fast retransmit and fast recovery algorithms as defined in RFC-2001 [109]. The Full-Featured TCP configuration uses Selective Acknowledgments (SACK), defined in RFC-2018 [84], improved initial window settings, defined in RFC-2414 [10], TCP extensions for high performance from RFC-1323 [64], and has a slightly smaller MSS (due to the use of timestamps) to fit in the 1500 bytes that are used as the wireless LAN service data unit. Another important aspect is the buffer space on the network interface of the AP which contains all packets to be transmitted; a vast amount of packets may queue-up and overflows have a major impact on the results. In our simulations the AP buffer was set sufficiently large to avoid packet drops.

We have conducted extensive experimentation to validate our model. Table 2.2 summarizes the simulation settings specific to our experiments. Simulation runs were performed for two different TCP configurations, up to five file-size distributions, for two mean file size settings and up to seven load values. Each run was executed until a sufficiently small 95% Confidence Interval (CI) was obtained, often requiring durations up to 1000 hours of real time simulation for higher load values. Each of the outcomes is based on averages in the order of 2.6×10^6 samples (excluding an extensive warm-up period of up to 1.4×10^5 observations). In total, over 190 runs were performed for our validation, some of which have consumed up to 299 hours of computing time on state-of-the-art simulation servers. The results for several representative examples are outlined below.

Variable	Setting
X_{FTPget}	4096 bits
$X_{FTPclose}$	64 bits
TCP_{stack}	Reno, Full-Featured
X_{MSS}	11680 bits (Reno), 11584 bits (Full-Featured)
$X_{tcp/ip}$	320 bits (Reno), 416 bits (Full-Featured)
w	70080 bits (8760 bytes)
X_{file}	16×10^5 bits (2×10^5 bytes), 8×10^6 bits (10^6 bytes)
$\frac{1}{\lambda_{200kb}}$	$\{0.48, 0.46, \ldots, 0.38\}$ seconds
$\frac{1}{\lambda_{1Mb}}$	$\{2.4, 2.3, \ldots, 2.0, 1.9, 1.85\}$ seconds

Table 2.2: Specific simulation and model parameters.

2.3.2 Comparison

Figure 2.2 shows the expected file download time as a function of the load for (1) the $M/G/1$-PS-model (2) our analytic model presented in Section 2.2.3, and (3) simulations using exponential file size distributions.

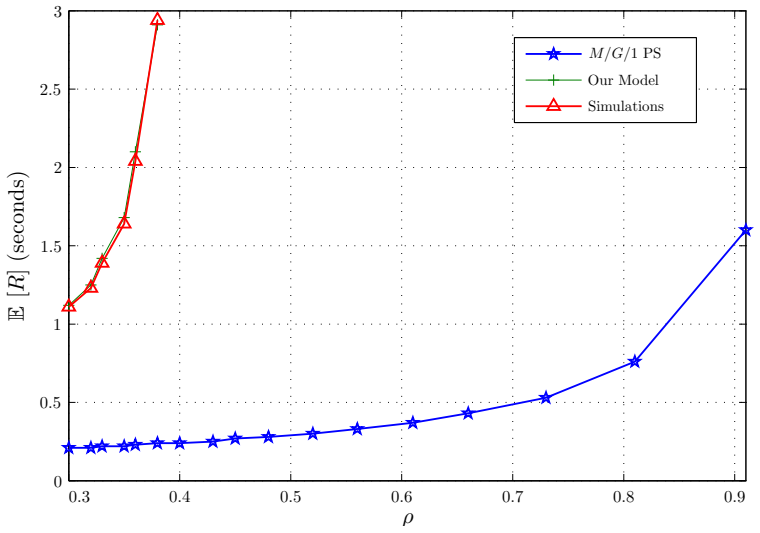


Figure 2.2: Expected download response time, $\mathbb{E}[R]$, as a function of the load, ρ, for various analytic models together with the simulation outcomes for the exponential distribution (using mean file size of 200 kBytes and Full-Featured TCP configuration); the *M/G/1*-PS model uses directly the load ($\lambda\beta$), *our model* corresponds to the model using the effective load-based model proposed in this chapter.

The results in Figure 2.2 illustrate the necessity for using the notion of effective

load, or a similar instrument to capture the medium efficiency that can be used in the $M/G/1$-PS model. The outcomes of our analytic model closely approximate the simulation outcomes depicted in Figure 2.2 and can hardly be distinguished. To illustrate the accuracy of our model we have, first, reproduced the models from Roijers et al. [106] and from Sakurai and Hanley [108], to match with the original outcomes specified in their contributions. Subsequently, we have parameterized the WLAN MAC models in accordance with one of our simulation settings (using long PHY preamble, TCP delayed acknowledgments and a TCP MSS of 1448 bytes). Finally, the obtained MAC-layer throughput is used in a $M/G/1$-PS model, but in contrast we use Poisson arrivals instead of applying admission control on the number of flows in the network [106], or admitting stations to have only one TCP flow [108]. In Figure 2.3 the expected file download time as a function of the effective load, ρ_{eff}, is shown for various models: (1) the analytic model from [106] (indicated as *RBF*), (2) the analytic model from [108] (*SH*), (3) our analytic model presented in Section 2.2.3 (*Our Model*), (4) a simplified version of our model (*Basic Model*) that ignores all detailed overhead due to the additional FTP/TCP interactions and transmitting the last cycle, and (5) simulations using exponential file-size distributions.

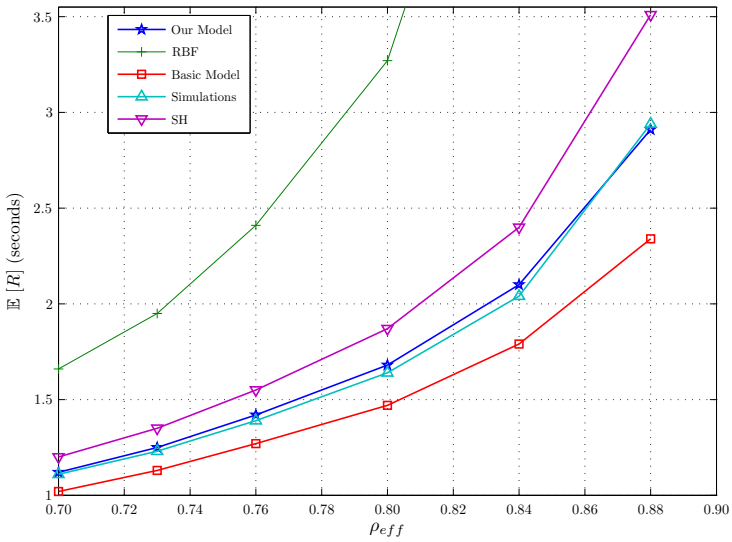


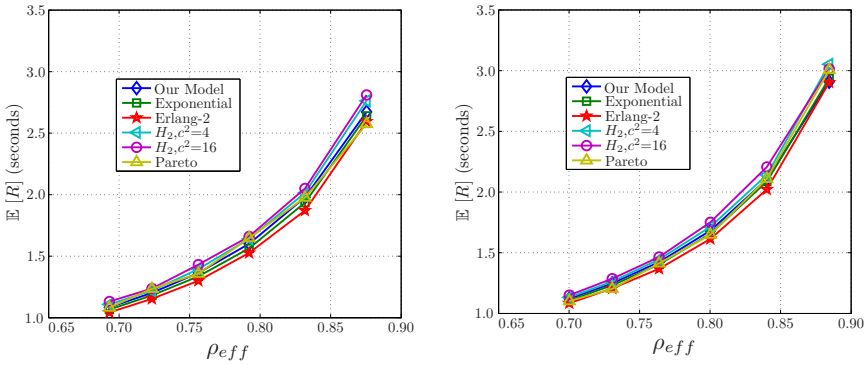
Figure 2.3: Average file download response time, $\mathbb{E}[R]$, as a function of the effective load, ρ_{eff}, for various analytic models together with the simulation outcomes for the exponential distribution (using mean file size of 200 kBytes and Full-Featured TCP configuration).

The results in Figure 2.3 lead to a number of interesting observations. First, our analytic model matches very closely with the simulation results with an error of $1 - 3\%$. Second, the results strongly outperform those from [106] (indicated

as RBF) in which the obtained file download times are severely over-estimated. The outcomes from [108] (indicated as SH) provide a closer match compared to those from [106], with an error of approximately 8% for an effective load of 0.70. For this load value the errors are far less sensitive to the obtained WLAN MAC throughput from the model than for 0.88, where relatively small differences in obtained WLAN throughput will cause large errors of approximately 20%. It should, however, be noted that the modeling assumption in [108] of having only one TCP flow per station is not respected in our validation and that our resulting effective load of approximately 0.9 is much larger than the 0.26 and 0.31 considered by [108] that is using traffic sources based on idle periods rather than Poisson arrivals. A third observation made from Figure 2.3 is that the model outcomes are inaccurate when the overhead associated with TCP and FTP session set-up and closure is ignored (see also Remark 2.5.3 below). Indeed, for an effective load of 0.70 fair outcomes are obtained with an error of 8%. However, when the effective load is increased to 0.88 the error becomes larger than 20%. These observations confirm our expectation, formulated in Section 2.2, that our model leads to more accurate outcomes for the circumstances we consider in this chapter.

2.3.3 Simulation results

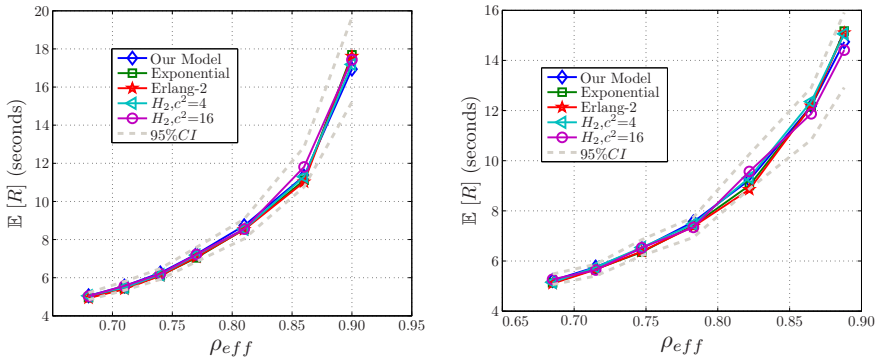
Our analytic model, defined in Section 2.2, inherits the insensitivity property of the mean response times with respect to the file-size distributions from the $M/G/1$-PS model. This raises the question to what extent the real download times are indeed insensitive. Figures 2.4 and 2.5 show the average file download response time, $\mathbb{E}[R]$, as a function of the effective load, ρ_{eff}, for a range of parameter combinations, including light- and heavy-tailed file-size distributions for different mean values and light- and heavy-load scenarios. We observe that the analytic results closely match those from the simulation for a wide range of model parameters. The results also suggest that there is sensitivity with respect to the file-size distribution, but this sensitivity is quite weak (with errors typically of 1 - 3% and no more than 8%). This observation is also in line with the one in Roberts [105]. However, it should be noted that perfect fairness will not be obtained. TCP's slow-start mechanism causes bias against small file sizes; during the slow-start phase TCP connections do not obtain a fair share of the medium capacity. File-size distributions with a higher variability will also receive greater influence because small files are more predominant (on which the TCP slow-start delay has relatively more impact). Despite this biased behavior of TCP, the influence on the simulation results is rather limited.



(a) Using TCP Reno implementation.

(b) Using Full-Featured TCP configuration.

Figure 2.4: Average file download response time, $\mathbb{E}[R]$, as a function of the effective load, ρ_{eff}, using exponential, Erlang-2, Pareto (with shape parameter = 1.33) and hyper-exponential file-size distributions with a mean value of 200 kBytes. The obtained 95% Confidence Intervals (CI) are at most 5.7% for the Pareto distribution and less than one percent for the other distributions.



(a) Using TCP Reno implementation.

(b) Using Full-Featured TCP configuration.

Figure 2.5: Average file download response time, $\mathbb{E}[R]$, as a function of the effective load, ρ_{eff}, using exponential, Erlang-2 and hyper-exponential file-size distributions with a mean value of 1 MByte. The 95% CI values are shown for the hyper-exponential (H_2, with $c^2 = 16$) file-size distribution which provides the largest ones observed.

To assess the impact of the file-size distribution on the AP buffer content distribution, Figure 2.6 shows the CCDF (on a log-scale), obtained both from sim-

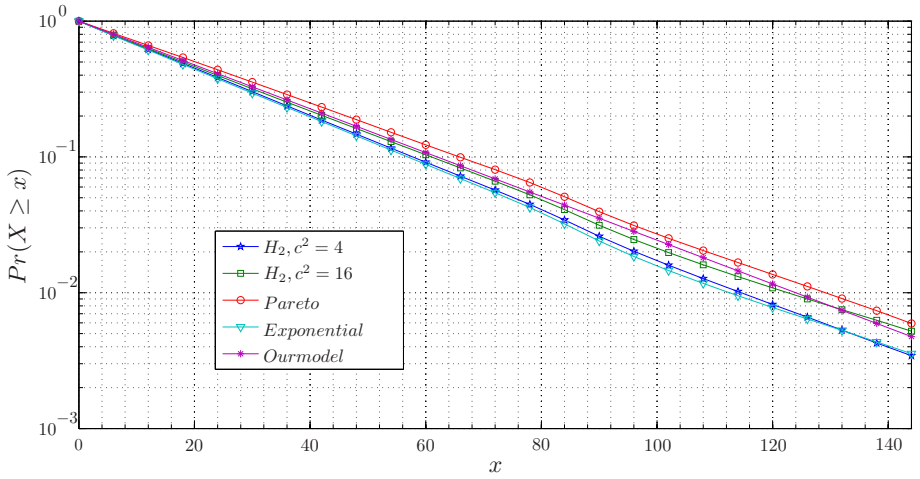


Figure 2.6: CCDF of the WLAN AP buffer content (x, in packets) from model and simulations, using exponential, Pareto (with $c^2 = \infty$) and hyper-exponential file-size distributions for the same mean file size and TCP configuration as in Figure 2.4b with $\rho_{eff} = 0.8$.

ulation, and from the analytic model using (2.33) under the assumption that condition (2.31) is satisfied (as is the case for the simulation settings outlined in Table 2.2).
The results suggest that there is no significant dependence between the AP buffer content distribution and the file-size distribution. Moreover, the results show that our analytic model accurately predicts the simulation outcomes. See Remark 2.5.2 in Section 2.5 for some additional comments on the AP buffer size.

2.4 Model validation by testbed experiments

2.4.1 Experimental setup

This section summarizes the results of extensive testbed experiments. Our aim is to validate whether (1) our analytic model accurately predicts the outcomes from our testbed, and (2) whether the mean download time in our testbed is indeed not very sensitive to the file-size distribution, as suggested by the $M/G/1$-PS model and shown by the simulation experiments. To this end, we have connected two powerful Ubuntu 9.04 Linux-based PCs; one functioning as FTP server, and the other as FTP clients. These PCs are interconnected by two independent and similar access networks to measure the download response times of file transfers in each network separately and compare the obtained values with the expected file-download time from the analytic model. In the experimental setup, the PC with the FTP clients generates all download requests,

according to independent Poisson processes. It is important to state that with a larger number of WLAN client devices there is, in addition to the AP, typically only one station contending for the medium at the same time, as reported in [106] and observed during the experiments in [50]. This justifies the choice of using one client device for our downloads rather than a large population.

At the PC operating as FTP server, files are generated according to an exponential and a hyper-exponential distribution with high variability (with a squared coefficient of variation equal to four, $c^2 = 4$), each of which consisted of 40.000 different files with mean size 2×10^5 bytes that are retrieved in a random order by the FTP clients.

In our testbed environment, each wireless access network consists of a Linksys (WAP54G) AP and an Ethernet bridge (WET54G) of the same hardware and firmware version that are tuned to different non-overlapping frequency channels to avoid mutual interference. Both devices are connected by a Mini-Ciruits power splitter (ZB8PD-4-S) to obtain a reproducible radio frequency environment with low interference and small propagation delay, T_p. Nonetheless, influences of other equipment remain. Both APs use a modified firmware program, called OpenWrt, that is specifically designed for embedded devices such as residential gateways and routers. This firmware offers detailed WLAN-MAC configuration options. Table 2.3 summarizes the parameters that are specific for our testbed configuration, including the IEEE 802.11 MAC parameters that differ from those specified in Table 2.1. These parameters have been selected as similar as possible to those parameters used in the simulation validation.

Variable	Setting
$X_{ftp-pasv}$	48 bits
$X_{ftp-227}$	392 bits
$X_{ftp-retr}$	272 bits
$X_{ftp-150}$	704 bits
$X_{ftp-226}$	184 bits
X_{beacon}	584 bits
X_{MSS}	11680 bits
$X_{tcp/ip}$	320 bits
w	70080 bits (8760 bytes)
X_{file}	16×10^5 bits (2×10^5 bytes)
$\frac{1}{\lambda}$	$\{0.48, 0.46, \ldots, 0.38, 0.36, 0.35\}$ seconds
R_c	$11 \cdot 10^6$ bps
BI	100 Time Units (TUs)
T_p	10^{-9} seconds

Table 2.3: Specific testbed environment and model parameters.

In Table 2.3 it is shown that the TCP stack in our testbed environment is

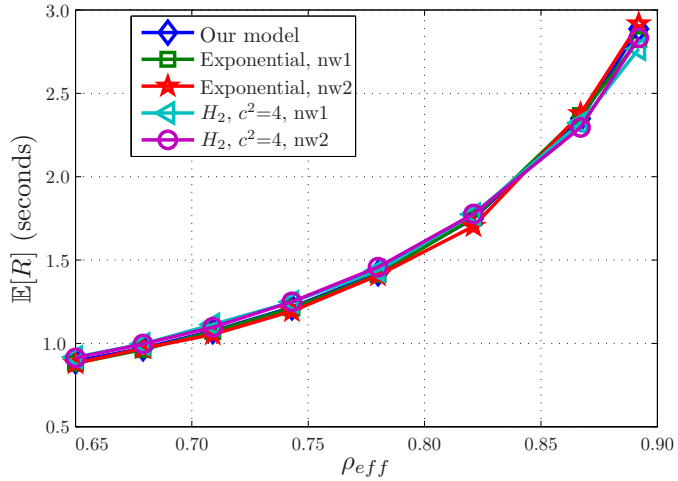


Figure 2.7: Analytic and experimental results, from network one and two, on the average download response time $\mathbb{E}[R]$ for exponential and hyper-exponential (H_2, with $c^2 = 4$) file-size distributions with an average of 2×10^5 bytes.

configured to use an MSS, indicated as X_{MSS}, of 1460 bytes and a window-size, w, of 8760 bytes. The standard TCP implementation from the Ubuntu 9.04 release is used, that is an implementation of the TCP protocol defined in RFC-793, RFC-1122 and RFC-2001 with the NewReno and SACK extensions. The TCP stack is configured accordingly to support the 8760-byte window-size and an MSS of 1460 bytes. Therefore we use in the analytic model 40 bytes of TCP/IP overhead of the higher-layer protocols, represented by variable $X_{tcp/ip}$. With regard to the IEEE 802.11 MAC parameters, slightly different settings are used in our experiments; (1) the AP is configured to broadcast beacon frames at a transmission rate, R_m, equal to 10^6 bps (specified in Table 2.1) with an interval of 100 time units of 1.024 microseconds each, and (2) control frames (WLAN acknowledgments) are transmitted at a rate, R_c, equal to the transmission of data frames, $11 \cdot 10^6$ bps. In addition, the FTP commands and responses introduced in Section 2.2.4 are also specified in Table 2.3.

2.4.2 Experimental results

We have conducted extensive experimentation to validate our model. Runs were performed for two different file-size distributions, for two networks and eight different load values. Runs have been executed until a small 95% Confidence Interval (CI) was obtained, requiring durations of over 55 hours to gather up to 450.000 observations per run. The representative outcomes of our experiments are outlined below of which the 95% CI is 2.7% or less.

ρ_{eff}	Analytic model	Number of observations	Network 1 Exponential	Network 2 Exponential	Network 1 H_2, c^2=4	Network 2 H_2, c^2=4
0.65	0.88	15×10^4	0.88	0.88	0.92	0.91
0.68	0.96	25×10^4	0.96	0.97	1.00	1.00
0.71	1.07	25×10^4	1.07	1.05	1.11	1.10
0.74	1.21	45×10^4	1.21	1.19	1.25	1.25
0.78	1.41	45×10^4	1.41	1.41	1.44	1.46
0.82	1.75	45×10^4	1.75	1.70	1.77	1.78
0.87	2.37	45×10^4	2.37	2.38	2.32	2.30
0.89	2.82	45×10^4	2.82	2.92	2.77	2.83

Table 2.4: Analytic and experimental $\mathbb{E}[R]$ for $X_{file} = 2 \times 10^5$ bytes.

The results in Table 2.4 and Figure 2.7 demonstrate that the analytic results closely match the outcomes from the experimental testbed for a broad range of model parameters with an average error of 1.7% up to a maximum of 4.1%. Indeed, there is no significant dependence of the mean download time in our experiments on the file-size distribution, which extends the applicability of this PS-modeling approach towards real network equipment. As expected, the experimental results of both networks are very similar, which will be a useful property in Chapter 6.

2.5 Engineering guidelines for practical application

Any approximation method, by definition, has parameter combinations where the results become less accurate. This raises the need for simple practical guidelines to determine for which combinations of parameters the model is valid. To this end, we have investigated the combined impact of the effective load, the TCP maximum window size, and the WLAN AP buffer size. The results are outlined below. We re-emphasize that the dynamics of the protocol stack, ranging from the application to the physical layer, and its interactions are extremely complex, while practice calls for simple guidelines, and therefore should be judged from that perspective.

For sufficiently large buffers, the TCP retransmissions are solely due to timeouts and not to buffer overflow. Extensive experiments reveal that good results are obtained as long as:

$$\rho_{eff} \leq 0.9. \tag{2.34}$$

When ρ_{eff} exceeds 0.9, TCP retransmissions start to have a noticeable influence, as the segment delay often exceeds the TCP timeout value due to severe queueing at the AP buffer (see also Figure 2.8 below). For smaller values of the AP transmission buffer size Q (in segments of size X_{MSS}), it is of interest to know for which *combinations* of Q and ρ_{eff} the model is applicable. Assuming that condition (2.31) is fulfilled, we approximate the packet loss probability, P,

as:

$$P \approx Pr\,(X \geq Q) = \rho_{eff}^{Q}, \tag{2.35}$$

where X is related to the number of downloads in progress N by approximation (2.32) and the equality follows directly from (2.33). This immediately leads to the following rule of thumb for applicability of the model:

$$\rho_{eff}^{Q} \leq \alpha. \tag{2.36}$$

Extensive simulations suggest that accurate outcomes are obtained for α in the range of 1%-3%. For larger values of α TCP retransmissions have a profound impact on the download response time due to buffer overflows.

To illustrate the behavior that is observed when the effective load is extremely high (and (2.34) is not respected), additional simulations have been conducted. Figure 2.8 shows the outcomes of our analytic model and the simulations for the case of exponential file-size distributions. We observe that when the effective load ρ_{eff} exceeds (say) 95% the results tend to become inaccurate. Nonetheless, note that in practice sustained extreme load values (in excess of 90%) hardly occur in well-dimensioned systems. This is in line with the observations of Ben Fredj et al. [17], stating that networks with such extreme load values are severely under-dimensioned and call for other approaches.

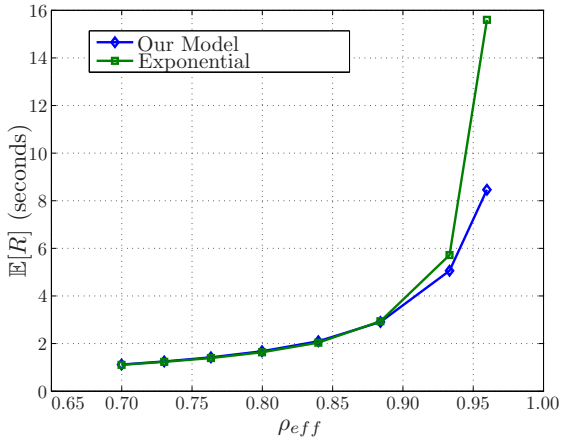
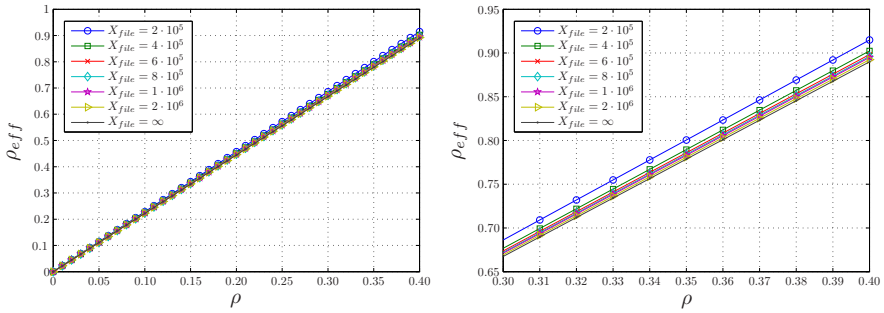


Figure 2.8: Average file download response time, $\mathbb{E}[R]$ from the analytic model and from simulations, as a function of the effective load, ρ_{eff} including overload values. The outcomes are obtained under the same conditions as for Figure 2.4b.

We end this section with a number of remarks.

Remark 2.5.1 (Effective load versus offered load)**.** The analytic model derived in Section 2.2.3 yields a linear relationship between the load, ρ, and the effective load, ρ_{eff}, on the network. Here, the load is defined by $\rho := \lambda\beta$ with λ the flow arrival rate and where β is the mean time to transmit files with mean size X_{file} on the WLAN medium at its transmission rate R. Note that R equals R_b for IEEE 802.11b-based systems and $R_{a/g/n}$ for IEEE 802.11a/g/n-based systems, respectively.



(a) From low to high effective load.

(b) From medium to high effective load.

Figure 2.9: Relationship in analytic model between load, ρ, and effective load, ρ_{eff}, for different values of the mean file size, X_{file} (in bits), under the same conditions as for Figure 2.4a.

Figure 2.9 shows the linear relation between the load of the effective load for various values of X_{file} that all meet in the origin. When X_{file} grows large, the slope converges to $T_{cycle}R/2X_{MSS}$, with R the transmission rate of the medium. This result is a direct consequence of (2.17). An intuitive explanation is that the influence of the TCP setup, closure and FTP commands on the response times tends to become negligible as the file size becomes large.

Remark 2.5.2 (AP transmission buffer)**.** For determining the AP buffer requirement using (2.36), condition (2.31) should be met. Otherwise an upper bound for the buffer size is obtained because the number of segments in flight will be lower than the number that fit in the maximum TCP window. It is important to note that the number of flows in the network is not limited, but geometrically distributed. This means that even for very large AP transmission buffers, overflows may occur. Furthermore, if condition (2.36) cannot be satisfied for the indicated range of α, the packet loss probability due to AP buffer overflow becomes substantial, model refinements taking TCP retransmissions explicitly into account are required, which opens up a challenging topic for further research.

Remark 2.5.3 (Applicability of the model to small file sizes)**.** Our PS-based analytic model presented in Section 2.2 implicitly assumes that the files are

transmitted in a sufficiently large number of TCP segments to justify the capacity sharing effect. In this context, Roberts [105] has indicated that the PS model does not capture all details of flows in networks that are controlled by TCP, where the slow-start algorithm causes a bias against small values of the file size and the additive increase multiplicative decrease introduces a bias against flows with high round-trip times. This limits the applicability of our model to file sizes that are representative for FTP traffic and requires several tens of TCP segments. For file sizes consisting of a few TCP segments only, the performance is dominated by the effects of round-trip times and TCP slow-start, which limits the applicability of the model for small file sizes.

Remark 2.5.4 (Relation to Bianchi-type of model)**.** Initially Bianchi's model [21], that uses a fixed-point approach, served as a basis for flow-level performance models on WLAN networks [76, 87]. As opposed to using (RTS/CTS) channel reservations, the aggregated WLAN throughput in basic access mode, when used without TCP, strongly depends on the number of active stations [21, 76]. Extensive validation shows that this is not the case when the data flows use TCP (as considered in this chapter) and are transferred over a WLAN that operates in basic access mode; the absence of admission control (either per station or for the whole network) has no significant impact on the performance of file transfers for a range of different parameter settings (as long as the requirements for the AP buffer and maximum effective load from (2.34) and (2.36) are respected). In fact, the simulation outcomes confirm that few stations contend for the medium at the WLAN MAC level. We observe that (1) the number of collisions on the medium is in line with what has been incorporated in the model (in (2.8) for instance), (2) the mean backoff interval drawn by all WLAN STAs is close to half of the *minimum* backoff window size (Cw_{min}), and (3) these two parameters do not depend significantly on the load (up to effective load values of approximately 0.9) nor the number of stations in the network. This implies that the medium contention is low, which reduces the need for including the fixed-point approach from [21] (in which all stations are assumed to always have a packet to transmit) for accurately modeling file transfers. Furthermore, large improvements in model accuracy are obtained when increasing the level of detail and applying correct parameterization to closely match outcomes of a commercially available simulation environment [91].

Remark 2.5.5 (TCP-stack implementations)**.** It should be noted that TCP-stack implementations vary from one operating system to another and may sometimes even allow user modifications for parameter tuning. In fact, recent operating systems, such as Windows Vista [86], have appeared that apply automatic parameter tuning, e.g. for setting the maximum receive window size to better utilize the available capacity on networks with a large bandwidth-delay product. These changes to the TCP-stack and the general trend towards creating larger receive window sizes (as proposed by [64]) may impact the performance and sensitivity of file transfers [87]. However, limited support of the window scaling techniques from [64] as well as the penetration rate of Vista may still lower the influence of these recent advances. Further work is needed

to assess the impact of automatic parameter adaptation techniques in TCP.

Remark 2.5.6 (Backward compatibility)**.** Practical WLAN deployments rely on the same IEEE 802.11 basic standard [1] for their fair method of medium access control, a property that is adopted by our proposed analytic model. In these circumstances stations may operate on different transmission rates or even on a different standard to support higher data rates or prioritization techniques. In this context we refer to [93] for a detailed description on the use of the IEEE 802.11 a/b/e/g/n standards and a mixture thereof with the influence of backward compatibility.

2.6 Conclusions and further research

For engineering purposes, there is a need for very simple, explicit, yet accurate, models that predict the performance of WLANs under anticipated load conditions. In this context, several detailed packet-level models have been proposed, based on fixed-point equations. Despite the fact that these models generally lead to accurate performance predictions, they do not lead to simple explicit expressions for the performance of WLANs.

The complex combined dynamics and protocol overhead of the 802.11 MAC, IP, TCP and application-layer can be captured into an explicit expression for a single parameter, called the *effective service time*. Based on the effective service time, the *effective load* can be defined to describe the flow-level performance of file transfers over WLANs with an $M/G/1$-PS model. The results show that the model leads to highly accurate predictions over a wide range of parameter combinations, including light- and heavy-tailed file-size distributions and light- and heavy-load scenarios. The simplicity and accuracy of the model make the results of high practical relevance and useful for performance engineering purposes.

The mapping from packet-level to the flow-level performance model based on combining the $M/G/1$-PS model with load ρ_{eff} enables us to use known analytic results for PS models. It is an interesting topic for further research to generalize our approach to formulating an effective service time and resulting effective load for a wider range of access networks. Furthermore, a wealth of literature is available on PS models, we refer to [114, 115, 24, 25] and references therein for results on sojourn times in PS models. For example, rather than focusing on the unconditional expected response times, see (2.29) and the simulation results, we may extend the results to the conditional expected response times, the conditional and unconditional variance, and in some cases even for the tail probabilities of the response times, based on known results for the $M/G/1$-PS model [33]. To assess the usefulness of such results is a challenging topic for further research.

In this chapter, the focus is on the performance of data transfers in the down-

load direction. However many applications generate large amounts of traffic in the upstream direction, which raises the question to what extent our analytic model is also applicable to networks where a mixture of up and download traffic is present. In the context of comparing the performance of FTP download and upload traffic, the authors in [108] have observed that the received throughput of upstream and downstream flows are "virtually identical" in circumstances that are similar to those considered in this chapter. This suggests that our model may also accurately predict the performance of traffic in the upstream direction, which is a challenging topic for further research.

Chapter 3

Tail-Optimal Static Traffic Splitting over Multiple Networks

Based on the model proposed in Chapter 2 a relationship can be established between TCP-based file transfers in a communication network and jobs that are processed in a PS-queue. Using this relationship, optimization problems may be investigated in a queueing theoretical context to subsequently apply the solutions found to communication networks.

The optimization problem considered in this chapter is one in which files are split upon arrival and subsequently assigned to parallel communication networks. When all networks have transferred their part of the file, the original file is re-assembled. The aim of splitting files and transferring the portions in parallel networks is to lower the time to transfer a file from a certain stream. Two types of streams are present in the considered system: one stream, called the foreground stream, of which the files are split and another type of stream, called background, of which the entire files are transferred by one network.

The splitting problem is studied in a queueing theoretical context, where jobs may be fragmented and assigned to different parallel PS-queues or are processed by one specific PS-queue without being split. We concentrate on the time to process a job and its fragments. More specifically our interest is in the sojourn time tail behavior of the foreground traffic. For the case of light foreground traffic and regularly varying foreground job-size distributions, a reduced-load approximation (RLA) is obtained for the sojourn times, similar to that of a single PS-queue. It is proven that a simple splitting rule provides tail-optimal performance with respect to the foreground sojourn time.

This result provides a theoretical foundation for the effectiveness of such a simple splitting rule. Extensive simulations demonstrate that this simple rule indeed performs well, not only with respect to the tail asymptotics, but also with respect to the mean sojourn times. The simulations further support our conjecture that the same splitting rule is also tail-optimal for non-light foreground traffic.

Finally, it is observed that the mean sojourn times are nearly-insensitive with respect to the file-size distribution.

This chapter is based on the results presented in [56].

3.1 Introduction

This chapter considers N parallel communication networks, each of which is modeled as a PS-queue that handles two types of traffic: foreground and background streams. The foreground traffic stream consists of jobs, each of which is split into N fragments according to a static splitting rule $(\alpha_1, \ldots, \alpha_N)$, where $\sum \alpha_i = 1$ and $\alpha_i > 0$ is the fraction of the job that is directed to PS-queue i. Upon completion of transmission of all fragments, it is re-assembled at the receiving end. The background streams use dedicated queues without being split.

An important implication of the obtained RLA is that the tail-optimal splitting rule is obtained by simply choosing α_i proportional to $c_i - \rho_i$, where c_i is the capacity of PS-queue i and ρ_i is the load offered to PS-queue i by the corresponding background stream.

In [42], the authors investigate the same queueing model in the context of web-server farms. A slight difference is that they do not consider the presence of background streams. The major difference is that they analyze the routing policy Join the Shortest Queue (JSQ) while this chapter concentrates on a splitting policy. Note that as opposed to communication networks, splitting in the context of web-server farms is not always possible. Two other related papers are [74] and [75]. In these papers the author considers a similar network but with FCFS queues and with probabilistic splitting. The reader is furthermore referred to [11], where the authors consider routing policies of the model in a distributed vs. centralized optimization. In general the queueing model considered in this chapter falls within the framework of a fork-join queueing network [12]. To the best of the author's knowledge such a queueing network in which nodes are PS-queues has not been investigated.

The rest of this chapter is organized as follows. In Section 3.2 the splitting problem and a simple splitting rule are described. Section 3.3 provides a heuristic derivation of the proposed splitting rule. In Section 3.4 it is proven that for light foreground traffic our splitting rule achieves tail asymptotic optimality. In Section 3.5 the simulation results are presented. These results are in agreement with the proposed conjecture. They further show "near insensitivity" with respect to the job size distributions and that in the case of light-tailed foreground job sizes the result does not hold. In Section 3.6 the relation between minimization of expected sojourn times and minimization of tails is discussed.

3.2 Problem formulation

The queueing model that is considered is the *concurrent access job-split* model, see Figure 3.1. There are N PS-queues that serve $N+1$ streams of jobs. Stream 0 is called the foreground stream and streams $1, \ldots, N$ are called the background streams. Jobs of background stream i are served exclusively at PS-queue i. Each job of the foreground stream is fragmented (split) upon arrival according to a static, splitting rule $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ where $\sum_{i=1}^{N} \alpha_i = 1$ and $\alpha_i \geq 0$, $i = 1, \ldots, N$. After splitting, the fragments are routed to their corresponding queues. Thus, when a job of size B arrives at stream 0, a fragment of size $\alpha_i B$ is directed to each queue i. Once all fragments complete their service, the fragments are reunited, and this completes the processing of the job that models the file transfer through the N communication networks.

Consider a tagged job of the foreground stream that arrives to a network in steady-state. Denote the sojourn time of its i'th fragment operating under the splitting rule $\underline{\alpha}$ by $V_i^{\underline{\alpha}}$. This is the time it takes the fragment to complete service at queue i. Denote $\underline{V}_{\underline{\alpha}} = (V_1^{\underline{\alpha}}, \ldots, V_N^{\underline{\alpha}})$. The sojourn time of the job through the network is $M_{\underline{\alpha}} = \max \underline{V}_{\underline{\alpha}}$. The objective is to analyze the distribution of $M_{\underline{\alpha}}$ and choose a splitting rule $\underline{\alpha}$ such that $M_{\underline{\alpha}}$ is kept minimal. The probabilistic and load assumptions are as follows. Arrivals of jobs in all streams are according to independent Poisson processes with rates $\lambda_i, i = 0, 1, \ldots, N$. Job sizes (representing the size of the file) of stream i constitute an i.i.d. sequence of positive random variables with finite expectation. The $N + 1$ sequences of job sizes are mutually independent. Denote the mean job size of stream i by β_i and $\rho_i = \lambda_i \beta_i$ $i = 0, 1, \ldots, N$. It is assumed that PS-queue i operates at rate c_i. For the background streams and queues, denote the corresponding N dimensional vectors $\underline{\rho}$ and $\underline{c}$. It is furthermore assumed that $\rho_0 \mathbf{1} + \underline{\rho} < \underline{c}$. Here $\mathbf{1}$ denotes a vector of 1's. This condition ensures stability irrespective of the choice of splitting proportions.

The splitting rule α^*

When background traffic occupies a certain fraction of each queue's capacity, the remaining capacity at each queue is available for serving the foreground traffic stream. Therefore the following splitting rule is defined that divides foreground jobs in parts that are proportional to the remaining capacity in each queue:

$$\alpha_i^* := \frac{c_i - \rho_i}{\sum_{j=1}^{N} (c_j - \rho_j)} \quad (i = 1, \ldots, N), \tag{3.1}$$

to obtain a splitting rule for all queues denoted by:

$$\underline{\alpha}^* := (\alpha_1^*, \ldots, \alpha_N^*). \tag{3.2}$$

Note that $c_i - \rho_i$ is the unutilized capacity of queue i due to background traffic and $\sum_{j=1}^{N} (c_j - \rho_j)$ is the total unutilized capacity due to background traffic.

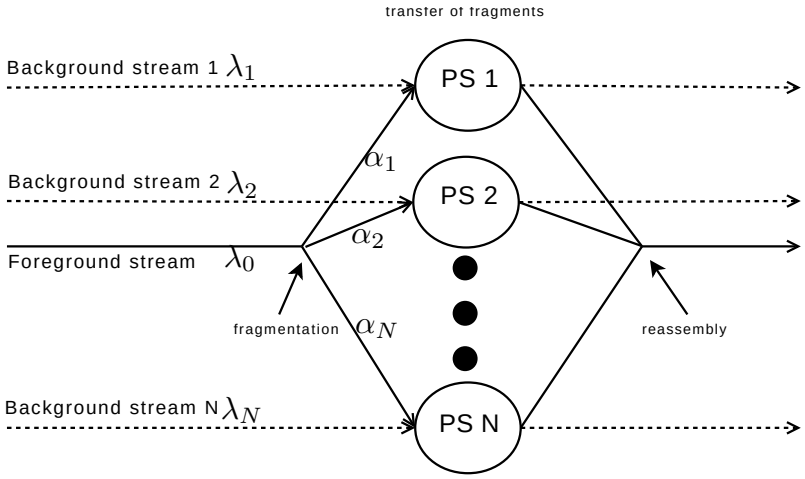


Figure 3.1: The concurrent access job-split model, where each of the N PS-queues represents a communication network that handles foreground and background file download traffic. In the concurrent access model, file downloads are modeled as jobs. Specifically for the job-split model, foreground jobs are fragmented into N tasks, which are subsequently processed in parallel by the N PS-queues and reunited into the original job when all fragments have been successfully processed. Background jobs are processed by one PS-queue exclusively.

Observe that α_i^* and $\underline{\alpha}^*$ do not depend on ρ_0. To motivate this rule, consider the following heuristic argument: Observe that each queue in isolation is a two class $M/G/1$-PS queue, allowing us to compute means. It is well known (first shown in [70]) that the mean sojourn time of a job of size B in a PS-queue with rate $\tilde{c}$ and load $\tilde{\rho}$ is:

$$\mathbb{E}[\tilde{V}|B] = \frac{B}{\tilde{c} - \tilde{\rho}}.$$

Assume now for simplicity that $N = 2$ and set $\alpha := \alpha_1$ and $(\alpha_2 := 1 - \alpha)$. Upon arrival of a foreground job of size B the expected sojourn time for each of the tasks can be written as:

$$\mathbb{E}[V_1|B] = \frac{\alpha B}{c_1 - \rho_1 - \alpha\rho_0}, \quad \mathbb{E}[V_2|B] = \frac{(1-\alpha)B}{c_2 - \rho_2 - (1-\alpha)\rho_0}.$$

Equating the above quantities and solving for α leads to (3.2). For the theoretical results, it is further assumed that the distribution of stream 0 files is regularly varying of index $\nu > 1$. This means that the tail of the distribution function has the form $Pr(B > x) = L(x)x^{-\nu}$, where $L(\cdot)$ is a slowly varying function: $L(ax)/L(x) \to 1$ as $x \to \infty$ for any $a > 0$. The background job

sizes do not have to be heavy-tailed, but there should exist an $\epsilon_i > 0$ such that $\mathbb{E}[B_i^{1+\epsilon_i}] < \infty$, where B_i denotes a generic random variable representing the job size of background stream i. Denote,

$$\gamma_{\underline{m}}^{\underline{\alpha}} := \min_{i=1,\ldots,N} \left(\frac{c_i - \rho_i}{\alpha_i} \right) - \rho_0. \tag{3.3}$$

A key result is:

$$Pr(M_{\underline{\alpha}} > x) \sim Pr(B > \gamma_{\underline{m}}^{\underline{\alpha}} x). \tag{3.4}$$

Here $f(x) \sim g(x)$ implies that $\lim_{x\to\infty} f(x)/g(x) = 1$. This is a form of a *Reduced Load Approximation* (RLA) (cf. [41], [24]) which appears in the considered network. It is further evident that in this case, the splitting rule which maximizes $\gamma_{\underline{m}}^{\underline{\alpha}}$ is $\underline{\alpha}^*$ and thus tail asymptotic optimality is achieved:

$$\limsup_{x\to\infty} \frac{Pr(M^{\alpha^*} > x)}{Pr(M^{\alpha} > x)} \leq 1, \quad \text{for all splitting rules } \underline{\alpha}. \tag{3.5}$$

This tail asymptotic optimality of the design parameter $\underline{\alpha}^*$ is similar to the tail-optimality properties of scheduling disciplines discussed in [26]. In this chapter a proof is presented of (3.4) for the case of *light foreground traffic*. In this case $\lambda_0 = 0$, which motivates the assumption that a single foreground job arrives to a steady state system. It is further conjectured that (3.4) is true for the general case. Extensive simulation experiments support this conjecture.

3.3 Heuristic derivation of the proposed splitting rule

Denote by B a random variable distributed as the job size of the foreground traffic stream. Denote by $Q_i^{\underline{\alpha}}(t)$ the number of files in queue i at time t, operating under a splitting rule $\underline{\alpha}$. Define,

$$R_i^{\underline{\alpha}}(x) := \int_0^x \frac{1}{1 + Q_i^{\underline{\alpha}}(t)} dt,$$

as the amount of service that a permanent customer obtains in queue i during the time $[0, x]$ when operating under the splitting rule $\underline{\alpha}$. Further denote by $\underline{R}^{\underline{\alpha}}(x)$ the N dimensional vector of $R_i^{\underline{\alpha}}(x)$. The following is obtained:

$$\begin{aligned} Pr(M_{\underline{\alpha}} > x) &= 1 - Pr(M_{\underline{\alpha}} \leq x) \\ &= 1 - Pr(\underline{V}_{\underline{\alpha}} \leq x\mathbf{1}) \\ &= 1 - Pr(B\underline{\alpha} \leq \underline{R}^{\underline{\alpha}}(x)). \end{aligned} \tag{3.6}$$

The first and second equalities are trivial. The third equality is due to the fact that in a PS-queue $Pr(\tilde{V} > \tilde{x}) = Pr(\tilde{B} > \tilde{R}(\tilde{x}))$. Observe now that,

$$\lim_{x\to\infty} \frac{1}{x} \underline{R}^{\underline{\alpha}}(x) = \underline{c} - \underline{\rho} - \rho_0 \underline{\alpha} \quad \text{a.s..}$$

As a consequence, since for large x, $\underline{R}^{\underline{\alpha}}(x) \approx (\underline{c} - \underline{\rho} - \rho_0\underline{\alpha})x$, it is anticipated that for a large x:

$$Pr(B\underline{\alpha} > \underline{R}^{\underline{\alpha}}(x)) \approx Pr(B\underline{\alpha} > (\underline{c} - \underline{\rho} - \rho_0\underline{\alpha})x). \tag{3.7}$$

Here the N dimensional random process $\underline{R}^{\underline{\alpha}}(x)$ is replaced by its asymptotic value. Heuristically, such an equivalence should hold when $\underline{R}^{\underline{\alpha}}(x)/x$ converges fast compared to the decay of the tail of B. In the next section it is proven that this relationship holds in the light foreground traffic case and conjectured to also hold in the general case.

Assuming (3.7) to be true and continuing heuristically from (3.6) the following is obtained:

$$\begin{aligned} Pr(M_{\underline{\alpha}} > x) &\approx 1 - Pr(B\underline{\alpha} \leq (\underline{c} - \underline{\rho} - \rho_0\underline{\alpha})x) \\ &= 1 - Pr(B \leq \min_{i=1,\ldots,N} \left(\frac{c_i - \rho_i - \rho_0\alpha_i}{\alpha_i} \right) x) \\ &= Pr(B > \gamma_m^{\underline{\alpha}}\, x). \end{aligned}$$

Where $\gamma_m^{\underline{\alpha}}$ is given by (3.3), which leads to the reduced load approximation (3.4). Observe now that maximizing $\gamma_m^{\underline{\alpha}}$ minimizes $Pr(B > x\,\gamma_m^{\underline{\alpha}})$ for any x. Finding the tail-optimal $\underline{\alpha}$ can be done by solving the following optimization problem:

$$\begin{aligned} \max_{\underline{\alpha}} \quad & \min_{i=1,\ldots,N} \left(\frac{c_i - \rho_i}{\alpha_i} \right) \\ \text{such that} \quad & \sum_{i=0}^{N} \alpha_i = 1 \\ & \underline{\alpha} > \underline{0}. \end{aligned} \tag{3.8}$$

It is clear that an optimizer of (3.8) achieves the tail asymptotic optimality (3.5). It is now shown that this solution is found to be $\underline{\alpha}^*$ as in (3.2).

Lemma 1. *The unique solution of (3.8) is given by $\underline{\alpha}^*$ from (3.2).*

Proof. For clarity denote $f_i = c_i - \rho_i$. Denote by $\underline{\alpha}'$ an optimal solution such that without loss of generality:

$$\frac{f_1}{\alpha_1'} \leq \ldots \leq \frac{f_N}{\alpha_N'}.$$

Observe that under α^*, the objective function is $\sum_{j=1}^{N} f_j$. Thus optimality of α' yields:

$$\frac{f_i}{\alpha_i'} \geq \frac{f_1}{\alpha_1'} \geq \sum_{j=1}^{N} f_j,$$

or,

$$f_i \geq \alpha_i' \sum_{j=1}^{N} f_j \quad (i = 1, \ldots, N).$$

Summing over i leads to:

$$f_i = \alpha_i' \sum_{j=1}^{N} f_j \quad (i = 1, \ldots, N),$$

since the summands are non-negative. This shows that $\underline{\alpha}' = \underline{\alpha}^*$ is the unique optimal solution.

3.4 Reduced load equivalence

For ease of notation of this section, an arbitrary splitting rule is fixed and the subscript/superscript $\underline{\alpha}$ is removed from all variables defined previously. Denote,

$$\gamma_i := \frac{c_i - \rho_i - \alpha_i \rho_0}{\alpha_i},$$

and observe that as in (3.3), $\gamma_m = \min_{i=1,\ldots,N} \gamma_i$. The following lemma states conditions under which the RLA (3.4) holds for our model. It is a direct application of results from [127] and [41]. See [24] for a survey.

Lemma 2. *Assume that* $\gamma_i := \frac{c_i - \rho_i - \alpha_i \rho_0}{\alpha_i}$,

$$\max\left(\frac{R_1(x)}{\alpha_1 x}, \ldots, \frac{R_N(x)}{\alpha_N x}\right) \to \max(\gamma_1, \ldots, \gamma_N) \text{ a.s.}, \tag{3.9}$$

and that there exists a positive finite constant K_m *such that*

$$Pr\left(\max\left(\frac{R_1(x)}{\alpha_1}, \ldots, \frac{R_N(x)}{\alpha_N}\right) \leq \frac{x}{K_m}\right) = o\left(Pr\left(B > \max\left(\gamma_1, \ldots, \gamma_N\right) x\right)\right), \tag{3.10}$$

then the reduced load approximation (3.4) is obtained: $Pr(M > x) \sim Pr(B > \gamma_m x)$.

Proof. Each of the PS-queues is a multi-class queue with two classes: foreground and background. Since background job sizes have a $1 + \epsilon$ finite moment and foreground file sizes have a regularly varying distribution, Theorem 4.2 of [24] (originally from [127]) is applied to obtain:

$$Pr\left(B > \frac{R_i(x)}{\alpha_i}\right) \sim Pr\left(B > \gamma_i x\right), \ i = 1, \ldots, N. \tag{3.11}$$

Now using the assumptions (3.9) and (3.10) Theorem 1 of [41] is applied to obtain:

$$Pr\left(B > \max\left(\frac{R_1(x)}{\alpha_1}, \ldots, \frac{R_N(x)}{\alpha_N}\right)\right) \sim Pr\left(B > \max\left(\gamma_1, \ldots, \gamma_N\right) x\right). \quad (3.12)$$

The rest of the proof is for the case $N = 2$ (the general case is more tedious but not more complicated, it requires using the inclusion exclusion law for the union of N events). First observe:

$$\begin{aligned} Pr(M > x) &= Pr(V_1 > x \text{ or } V_2 > x) \\ &= Pr(V_1 > x) + Pr(V_2 > x) - Pr(V_1 > x, V_2 > x) \\ &= Pr(\alpha_1 B > R_1(x)) + Pr(\alpha_2 B > R_2(x)) \\ &- Pr(\alpha_1 B > R_1(x), \alpha_2 B > R_2(x)) \\ &= Pr\left(B > \frac{R_1(x)}{\alpha_1}\right) + Pr\left(B > \frac{R_2(x)}{\alpha_2}\right) \\ &- Pr\left(B > \max\left(\frac{R_1(x)}{\alpha_1}, \frac{R_2(x)}{\alpha_2}\right)\right). \end{aligned}$$

Now assume that $\gamma_1 \leq \gamma_2$ and thus $\gamma_m = \gamma_1$ and $\max(\gamma_1, \gamma_2) = \gamma_2$:

$$\begin{aligned} \frac{Pr\left(M > x\right)}{Pr\left(B > \gamma_m x\right)} &= \frac{Pr\left(B > \frac{R_1(x)}{\alpha_1}\right)}{Pr(B > \gamma_1 x)} \\ &+ \frac{Pr\left(B > \frac{R_2(x)}{\alpha_2}\right) - Pr\left(B > \max\left(\frac{R_1(x)}{\alpha_1}, \frac{R_2(x)}{\alpha_2}\right)\right)}{Pr(B > \gamma_1 x)} \\ &= \frac{Pr\left(B > \frac{R_1(x)}{\alpha_1}\right)}{Pr(B > \gamma_1 x)} \\ &+ \frac{Pr(B > \gamma_2 x)}{Pr(B > \gamma_1 x)} \left(\frac{Pr\left(B > \frac{R_2(x)}{\alpha_2}\right)}{Pr(B > \gamma_2 x)} - \frac{Pr\left(B > \max\left(\frac{R_1(x)}{\alpha_1}, \frac{R_2(x)}{\alpha_2}\right)\right)}{Pr(B > \max(\gamma_1, \gamma_2) x)} \right). \end{aligned}$$

Now,

$$\frac{Pr(B > \gamma_2 x)}{Pr(B > \gamma_1 x)} = \frac{L(\gamma_2 x)}{L(\gamma_1 x)} \left(\frac{\gamma_2}{\gamma_1}\right)^{-\nu} \to \left(\frac{\gamma_2}{\gamma_1}\right)^{-\nu},$$

and from (3.11) and (3.12) the result is obtained. The case of $\gamma_2 > \gamma_1$ is symmetric.

Now the RLA (3.4) can be supported and the asymptotic optimality of α^*. The result applies to the light foreground traffic case.

Theorem 3.4.1. *Consider the concurrent access job-split model in light foreground traffic: there is a single foreground arrival to steady state with $\lambda_0 = 0$. Then the reduced load approximation (3.4): $Pr(M_{\underline{\alpha}} > x) \sim Pr(B > \gamma \frac{\alpha}{m} x)$ holds.*

Proof. When applying Lemma 2: (3.9) follows from the law of large numbers. To see (3.10) observe that:

$$\begin{aligned} Pr\left(\max\left(\frac{R_1(x)}{\alpha_1},\ldots,\frac{R_N(x)}{\alpha_N}\right)\leq\frac{x}{K_m}\right) &= Pr\left(\frac{R_1(x)}{\alpha_1}\leq\frac{x}{K_m},\ldots,\frac{R_N(x)}{\alpha_N}\leq\frac{x}{K_m}\right) \\ &= \prod_{i=1}^{N} Pr\left(\frac{R_i(x)}{\alpha_i}\leq\frac{x}{K_m}\right) \end{aligned}$$

Where under the light foreground traffic assumption all queues are in steady state and there is a single arrival, thus $R_i(\cdot)$ are independent. Now as proved in [41] (Theorem 2), each of the terms can be made $o(Pr(B>x))$ by choosing K_m appropriately. Thus (3.10) is obtained.

Using this proof method to repeat the above for the non-light foreground traffic case requires more care in obtaining (3.9) and (3.10). It is conjectured that these conditions indeed hold and thus:

Conjecture 3.4.1. *Theorem 3.4.1 holds also in the non-light foreground traffic case and thus the splitting rule α^* is in general tail-optimal.*

In the next section simulation results are presented that support the validity of this conjecture.

3.5 Simulation results

This section summarizes the results of some extensive simulations for evaluating $Pr(M_\alpha>x)$ on some examples with $N=2$. For convenience we denote $\alpha:=\alpha_1$ $(1-\alpha=\alpha_2)$, similarly for α^*. With respect to the tail probabilities, the primary purpose is to assert Conjecture 3.4.1 and the behavior of the tail-optimality claim (3.5) by estimating,

$$\alpha^*(x)=\text{argmin}_\alpha Pr(M_\alpha>x),\quad \text{and}\quad P^*(x)=Pr(M_{\alpha^*(x)}>x).$$

In this respect, we attempt to observe graphically that $\hat{\alpha}^*(x)\to\alpha^*$ as $x\to\infty$, where the estimators are denoted by hats. To illustrate the relative sub-optimality for a finite x when using α^* instead of $\alpha^*(x)$ the following is plotted:

$$\frac{\hat{P}(M_{\alpha^*}>x)-\hat{P}^*(x)}{\hat{P}^*(x)}. \tag{3.13}$$

In general, obtaining such results by simulation requires long runs since we are trying to optimize probabilities of a rare event. In addition, the data of the simulation runs is used to analyze $\mathbb{E}[M_\alpha]$, show that it is nearly insensitive to the file size distributions and compare the splitting rule to the JSQ routing policy.

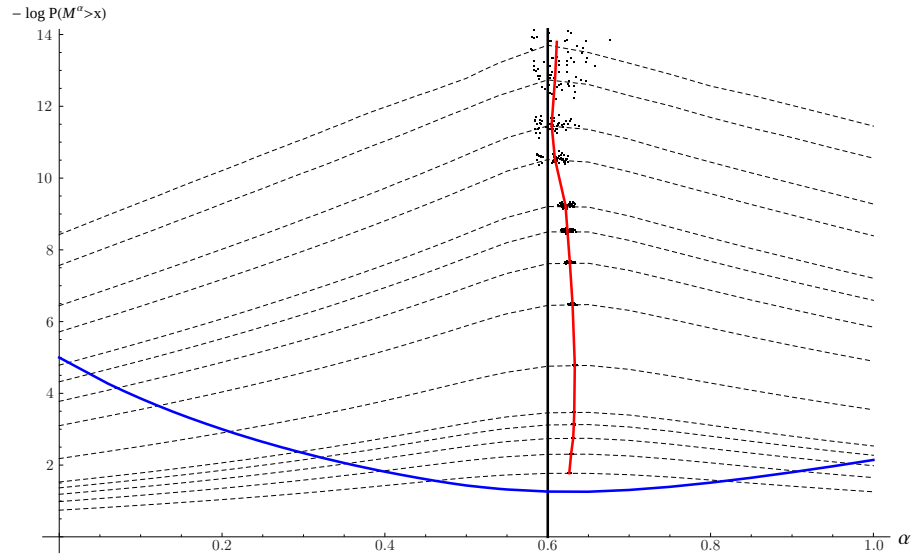


Figure 3.2: An illustration of the data analysis approach: System 4 as an example. Dashed curves are plots of estimates of $-\log\ Pr(M_\alpha > x)$ for $x = 1, 2, 3, 5, 8, 11, 17, 25, 35, 48, 64, 85, 115, 160, 210, 270, 350, 500$. These curves are maximized by the thick trajectory of $\alpha^*(x)$ which converges to the vertical line at $\alpha^* = 0.6$. Clouds of optimizers over the 50 repetitions are plotted in order to present the dispersion in the argmax estimates. The convex dotted curve is the estimate of $\mathbb{E}[M_\alpha]$ drawn on the same scale.

In all runs we set $\beta_0 = \beta_1 = \beta_2 = 1$ and $c_1 = c_2 = 1$. The types of file size distributions considered are deterministic, exponential, Erlang-2 (a sum of two i.i.d. exponentials) and Pareto-3 (which is regularly varying with index $\nu = 3$). Here the Pareto distribution is used with support $[0, \infty)$, i.e. $Pr(B > x) = (1 + x/2)^{-3}$. The simulation runs are further parameterized by the following:

$$\rho = \frac{\lambda_0 + \lambda_1 + \lambda_2}{2}, \quad \kappa = \frac{1 - \lambda_1}{1 - \lambda_2}, \quad \eta = \frac{\lambda_0}{\lambda_1 + \lambda_2}.$$

ρ is the total load on the system, κ is the ratio of free capacity and η is the ratio of foreground to background traffic. These 3 values uniquely define λ_0, λ_1 and λ_2. The table below specifies the parameters of the simulated systems.

System	ρ	κ	η	Distribution 0	Distribution 1	Distribution 2	$(\lambda_0, \lambda_1, \lambda_2)$	α^*
1	0.5	1.5	0.5	Pareto-3	Pareto-3	Pareto-3	$(\frac{1}{3}, \frac{1}{5}, \frac{7}{15})$	0.6
2	0.5	1.5	0.5	Pareto-3	Deterministic	Deterministic	as System 1	-
3	0.5	1.5	0.5	Pareto-3	Exponential	Exponential	as System 1	-
4	0.5	1.5	0.5	Pareto-3	Exponential	Deterministic	as System 1	-
5	0.5	1.5	0.5	Deterministic	Deterministic	Deterministic	as System 1	-
6	0.5	1.5	0.5	Erlang-2	Erlang-2	Erlang-2	as System 1	-
7	0.5	1.5	0.5	Exponential	Pareto-3	Erlang-2	as System 1	-
8	0.5	2.0	0.5	Pareto-3	Pareto-3	Pareto-3	$(\frac{1}{3}, \frac{1}{9}, \frac{5}{9})$	$\frac{2}{3}$
9	0.5	1.0	0.5	Exponential	Exponential	Exponential	$(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$	0.5

Table 3.1: Simulated systems.

Systems 1 through 7 all have the same rate parameters but vary in the job size distributions. System 8 is an additional example of an unbalanced system having $\kappa = 2.0$ and thus $\alpha^* = 2/3$. System 9 is a balanced system that is simulated for the sake of additional sanity checking: symmetric behavior of this system is expected.

Simulation runs are composed of 5×10^7 foreground jobs, starting empty. For each system the simulation is repeated for various values of α, using the same seed for all values. A fine grid is used of steps of 0.005 for α within the range of $[\alpha^* - 0.10, \alpha^* + 0.10]$. Outside of this range but within the range $[\alpha^* - 0.25, \alpha^* + 0.25]$ grid of steps of 0.02 are used. In the remaining region a grid of 0.05 is used. In addition, we ran each system using the Join the Shortest Queue (non-splitting) policy.

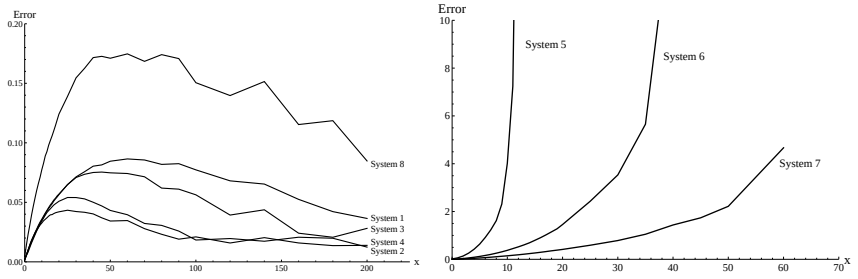
Per system the above specified range of α is repeated for 50 different seeds. Note that keeping the same seed while changing α is useful for optimizing the behavior of the queue given a single sample path of primitive job sizes over α. The total number of performed runs is about $30,000$ and the total number of foreground jobs that have passed through the simulated system is of the order of 1.5×10^{12}. The simulations use a short and efficient C program which we have coded.

3.5.1 Tail behavior

Figure 3.2 is a representative view of the obtained results. It is a plot of some of the data collected in the simulation runs of System 4. First, the tail probabilities $Pr(M_\alpha > x)$ are estimated for increasing values of x. These are plotted on a $-\log$ scale (dashed lines). These are subsequently optimized over α for increasing values of x. This yields the trajectory of $\hat{\alpha}^*(x)$ (thick curve). Clearly, as x grows the accuracy of this optimization is decreased due to the rarity of the tail event. We pictorially depict this in the figure by plotting the clouds of the 50 $(\text{argmax}_\alpha, \max_\alpha)$ pairs which result for increasing x's, one pair per seed. The thin vertical line in the figure is at $\alpha^* = 0.6$ and indeed, in agreement with the main conjecture and claim of this chapter, it appears as the limiting value

of $\alpha^*(x)$. An estimate $\mathbb{E}[M_\alpha]$ is plotted with a dot for every α in the grid. We comment on the mean in the next subsection.

Note that while Figure 3.2 shows that the argmax appears to converge rather slowly in x, it is more important to observe that the relative error (3.13) is always kept low. This can be observed in Figure 3.3a where (3.13) is plotted for the systems in which the foreground jobs have a heavy-tailed regularly varying service distribution. The same quantity for systems with light-tailed foreground jobs is plotted in Figure 3.3b. Here it appears the relative error explodes. Thus suggesting that α^* is not tail-optimal in the light-tailed foreground job size case. Note that the fact that tail-optimality of policies/rules is sometimes dependent on the tails of the primitive distributions also appears in other similar works. See for example [26] and [74].



(a) Heavy-tailed foreground job sizes. (b) Light-tailed foreground job sizes.

Figure 3.3: Graphs of (3.13), the relative distance from optimality for finite x: (a) Heavy-tailed foreground job sizes. (b) Light-tailed foreground job sizes.

3.5.2 Mean behavior

In Figure 3.4 the estimated values of $\mathbb{E}[M_\alpha]$ are plotted for systems $1-9$ for a range of α values. In addition, the values of α^* are marked for the various systems by vertical dashed lines and on these lines the mean sojourn times are dotted that are obtained for the systems using the JSQ routing policy. At α^*, the width of 99% confidence intervals for the mean (using 50 observations) are in the order of 10^{-4}. Some comments are due: First observe that in all these examples the following applies:

$$\mathbb{E}[M_{\alpha^*}] < \mathbb{E}[M_{\text{JSQ}}].$$

Secondly, observe that $\min_\alpha \mathbb{E}[M_\alpha] \approx \mathbb{E}[M_{\alpha^*}]$. This is a key result: The simple splitting rule proposed in this chapter (which is tail-optimal) is nearly optimal with respect to the mean. We further comment on this in the next section.

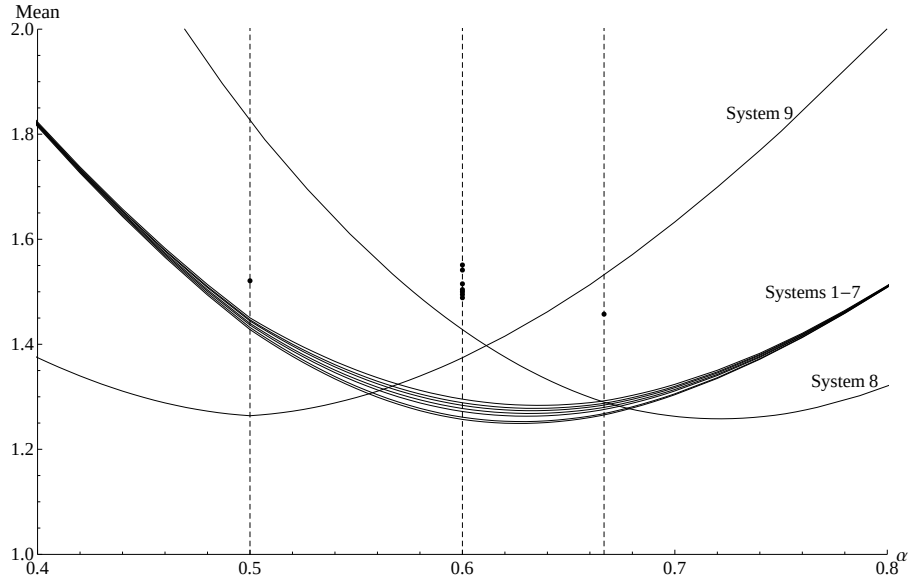


Figure 3.4: Mean sojourn time curves. Vertical lines are at $\alpha^* = 0.5, 0.6, 2/3$. Dots on the vertical lines are mean sojourn times using JSQ for the corresponding systems.

A third observation that appears from Systems $1-7$ is that the mean sojourn times (and mean queue sizes) are quite insensitive to the job size distribution. This property of JSQ was first observed and heavily investigated in [42] (for a system without background streams). Using the proposed splitting rule and taking $\alpha = 0$ or 1 yields two multi-class PS-queues which are known to be exactly insensitive (one of the two queues is single class). When $\alpha \neq 0, 1$ this is no longer the case, yet the figure shows that even when using $\alpha = \alpha^*$, the queues are "nearly insensitive". It is important to note that in [42] the authors show that not all routing policies have this "near insensitivity" property (even though a single PS-queue is insensitive). Note that the "magnitude" of the sensitivity of the splitting rule is similar to that of JSQ: The maximum difference in mean sojourn times due to the job size distribution is of the order of 4%.

3.6 Tail behavior versus mean behavior

Following Theorem 3.4.1 and Conjecture 3.4.1, α^* is a tail-optimal splitting rule. In addition, as observed in Figure 3.4 it nearly optimizes the mean. We now present two possible reasons for this relation between the optimization of the sojourn time tail and optimization of the mean sojourn time. Explanation 1 below is specific to the concurrent access job-split model and uses the asymptotic properties of the processes $R_i(x)$. Explanation 2 that follows presents a simple

general result regarding performance analysis of tails and means.

Explanation 1 Consider an arbitrary splitting rule α, and denote $R(x) := \min_{i=1,\ldots,N} \frac{R_i(x)}{\alpha_i}$. It can be observed that $\frac{R(x)}{x} \to \gamma_m$ and $\frac{R^{-1}(x)}{x} \to \frac{1}{\gamma_m}$, where the convergences are a.s. We have that $Pr(M > x) = Pr(B > R(x))$ and thus defining $M(b)$ as the sojourn time of a foreground job of size b, we have that $M(b) = R^{-1}(b)$. Define $\mu(b) := \mathbb{E}[M(b)]$. Since the underlying queue is regenerative, the almost sure convergence implies, $\frac{\mu(b)}{b} \to \frac{1}{\gamma_m}$ as $b \to \infty$. As a result, for large b:

$$\mu(b) \approx \frac{b}{\gamma_m}. \tag{3.14}$$

Thus selecting α such that γ_m is maximal minimizes $\mu(b)$ when b is large. It thus also approximately minimizes the unconditional sojourn time $\mathbb{E}[M] = \mathbb{E}_B[\mu(B)]$ where B is distributed as a foreground job size. Further observe that the relation (3.14) is similar to the distinctive feature of a standard PS-queue where the approximate equality is exact. This property also sheds light on the near insensitivity of our system since for large b it behaves similarly to a PS-queue.

Another observation is that the splitting rule α^* ensures that $\mathbb{E}[V_i]$ are equal. It is known that $\mathbb{E}[M] \geq \mathbb{E}[V_i]$ and also for a job of size b, we have $\mathbb{E}[M(b)] \geq \mathbb{E}[V_i(b)]$. The auxiliary results obtained for the reduced load equivalence suggest that, especially for large jobs, $\mathbb{E}[M(b)]$ and $\mathbb{E}[V_i(b)]$ are not too far apart.

Explanation 2 Consider an arbitrary stochastic model parameterized by α. Assume that the choice of α induces a non-negative distribution $1-\overline{F}_\alpha(x)$ with mean μ_α. For simplicity assume that α is scalar and that $1-\overline{F}_\alpha(x)$ is absolutely continuous. In the case of our model (for $N = 2$), $\alpha = \alpha_1$ and the distribution is that of the sojourn time.

Lemma 3. *Assume that $\overline{F}_\alpha(x)$ is unimodal in α and that $\overline{F}_\alpha(x)$ and μ_α are differentiable in α, then there exists an $x > 0$ such that*

$$argmin_\alpha \mu_\alpha = argmin_\alpha \overline{F}_\alpha(x).$$

The above result may be observed in Figure 3.2 where the trajectory of $\alpha^*(x)$, appears to cross the dotted $\mathbb{E}[M_\alpha]$ curve at its minimum. While typically finding the x at which these two curves cross, is difficult and not of practical importance, systems in which $\alpha^*(x)$ does not vary greatly in x will nearly optimize the mean when optimizing the tail. This appears to be the case in our system. Since $\alpha^*(x)$ trajectories do not vary greatly in x.

Proof. Denote by $\tilde{\alpha}$ a minimizer of μ_α. Denote $\mu'(\alpha) = \frac{d}{d\alpha}\mu_\alpha$. Then we have $\mu'(\tilde{\alpha}) = 0$. We also know that $\mu_\alpha = \int_0^\infty \overline{F}_\alpha(u)du$. Denote $\overline{F}'(\alpha, u) = \frac{d}{d\alpha}\overline{F}_\alpha(u)$ Combining the above we obtain,

$$0 = \int_0^\infty \overline{F}'(\tilde{\alpha}, u)du.$$

Thus $\overline{F}'(\tilde{\alpha}, u)$ is either constantly 0 or has to be both negative and positive and thus there must be a $\tilde{u}$ for which it equals 0. Thus since $\overline{F}_\alpha(x)$ is unimodal in α then for $x = \tilde{u}$ it is optimized by $\tilde{\alpha}$.

3.7 Conclusions and further research

For the case of light foreground traffic and regularly varying foreground job size distributions, a reduced-load approximation (RLA) for the sojourn times, similar to that of a single PS-queue can be obtained. An important implication of the RLA is that the tail-optimal splitting rule is simply to choose α_i proportional to $c_i - \rho_i$, where c_i is the service rate of queue i (that represents the capacity of network i) and ρ_i is the load offered to queue i by the corresponding background stream. This result provides a theoretical foundation for the effectiveness of such a simple splitting rule. Extensive simulations demonstrate that this simple rule indeed performs well, not only with respect to the tail asymptotics, but also with respect to the mean sojourn times. The simulations further support our conjecture that the same splitting rule is also tail-optimal for non-light foreground traffic. Finally, near-insensitivity of the mean sojourn times is observed with respect to the job-size distribution. It is an interesting topic for further research to investigate the tail-optimality for light-tailed foreground job size distributions as the simulations have indicated that the optimality of the proposed splitting rule in that case does not hold.

Chapter 4

Mean-Optimal Static Traffic Splitting over Multiple Networks

In the previous chapter tail-optimal traffic splitting is considered in a concurrent access job-split model where certain traffic streams are split and their fractions are assigned to different networks represented by PS-queues. It was observed that tail-optimal traffic splitting nearly optimizes the mean sojourn time of the foreground traffic stream. Analyzing the sojourn times of traffic streams that are split over multiple networks is difficult due to the complex correlation structure between the sojourn times at the PS-queues, which makes an exact detailed mathematical analysis of the model difficult. Therefore, in this chapter we propose a simple and accurate approximation for the splitting rule $\underline{\alpha}^*$ that minimizes the expected transfer time of file downloads that are split over N parallel communication networks. Our approximation is validated extensively by simulations. The results show that the outcomes are extremely accurate for a wide range of parameter combinations.

This chapter is based on the results presented in [53].

4.1 Introduction

In this chapter the same modeling abstraction is applied as in the previous chapter, where communication networks are modeled as a network of PS-queues that process jobs or fragments of jobs, corresponding to the transfer of files and portions thereof respectively. Similar to Chapter 3 static job-splitting rules $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ are studied, where a job of size τ is split into N tasks of size $\alpha_i\tau$ $(i = 1, \ldots, N)$, where the i-th task is processed by PS-queue i, and reassembled upon completion of all the tasks. In addition, we assume the presence of background traffic at each of the queues. Contrary to the previous chapter, the goal in the present chapter is to find a splitting rule $\underline{\alpha}^*$ that minimizes $\mathbb{E}[S_0^{\underline{\alpha}}]$, where $S_0^{\underline{\alpha}}$ is the total processing time of an entire foreground job, which generally depends on the job-size distributions and on the characteristics of the background traffic streams. Unfortunately, this model does not allow for an exact

analysis. The complexity lies in the fact that the sojourn times of the fragments in the different PS-queues are generally correlated. Therefore, we develop a new approximation for $\mathbb{E}[S_0^{\underline{\alpha}}]$, combining light- and heavy-traffic asymptotics, which then leads to an approximation for $\underline{\alpha}^*$. The approximation is validated by extensive simulations for a wide range of parameter combinations, including light- and heavy-tailed job-size distributions, and mixtures of light- and heavy-load scenarios on foreground and background traffic. These simulations demonstrate that the differences between the approximated optimal splitting rule and the estimated optimum with respect to the expected foreground sojourn time are extremely small for a wide range of parameter settings.

The organization of this chapter is as follows. In Section 4.2 the model is described and the notation is introduced. In Section 4.3 we analyze the performance of the model and use these insights to develop a new approximation method for determining the optimal split $\underline{\alpha}^*$. In Section 4.4 the accuracy of the approximation method is discussed in detail.

4.2 Model description

The queueing model we consider is the *concurrent access job-split* model that was presented in Section 3.1 and is depicted again in Figure 4.1. Similar to the previous chapter, files are modeled as jobs that may be fragmented into tasks. There are $N+1$ traffic streams: stream 0 is called the foreground stream and streams $1, \ldots, N$ are called the background streams. Jobs of background stream i are not fragmented and are served exclusively at PS_i $(i = 1, \ldots, N)$. Jobs of the foreground stream are fragmented into tasks upon arrival according to a fixed splitting rule $\underline{\alpha} = (\alpha_1, \ldots, \alpha_N)$ where $\sum_{i=1}^N \alpha_i = 1$ and $\alpha_i \geq 0$, $i = 1, \ldots, N$; thus a foreground job of size $B = \tau$ is split into N tasks of size $\alpha_i \tau$, and the i-th task is processed by PS_i $(i = 1, \ldots, N)$. Once all tasks have been completed, they are reassembled, which completes the processing of the job. In accordance with the model in Chapter 3, the arrivals of jobs in all streams are according to independent Poisson processes with rates $\lambda_i, i = 0, 1, \ldots, N$. The total arrival rate is denoted by $\Lambda = \lambda_0 + \lambda_1 + \cdots + \lambda_N$. For all streams, each job size B is an independent sample from a general distribution with finite k-th moment $\beta^{(k)} = \mathbb{E}[B^k]$, for $k = 1, 2, \ldots$.

For the background streams and queues, the corresponding N-dimensional vectors are denoted $\underline{\rho}$ and $\underline{c}$. It is furthermore assumed for stability that $\rho_0 \mathbf{1} + \underline{\rho} < \underline{c}$, with $\underline{c} = \mathbf{1}$. Here $\mathbf{1}$ denotes a vector of 1's.

Denote the background load of stream i by $\rho_i = \lambda_i \beta^{(1)}$ $(i = 0, 1, \ldots, N)$, and denote the total load offered to the system by $\rho = \rho_0 + \rho_1 + \cdots + \rho_N$. The utilization of queue i is denoted by

$$\xi_i := \rho_i + \alpha_i \rho_0, \text{ and let } \xi := \max_{i=1,\ldots,N} \xi_i. \tag{4.1}$$

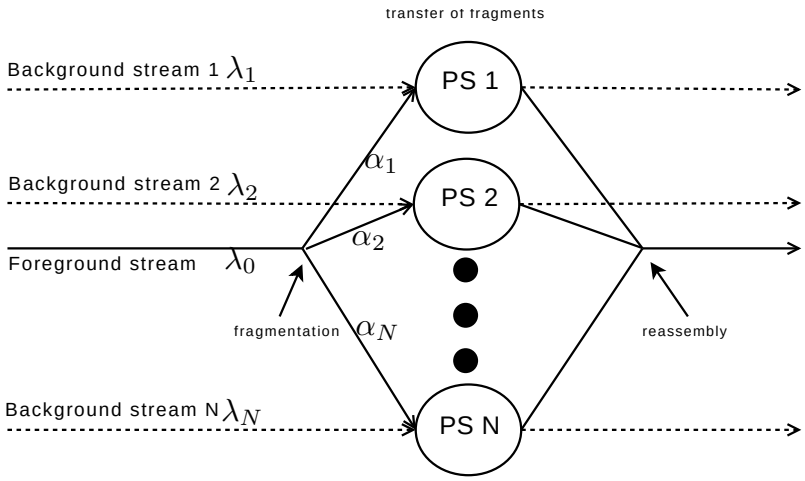


Figure 4.1: The concurrent access job-split model.

For stability, it is assumed that $\xi < 1$. Note that if $\rho > 1$ the stability condition may pose restrictions on the choice of the splitting rule $\underline{\alpha}$. Denote by $A := \{\underline{\alpha} : \xi < 1\}$, i.e., the set of combinations for which the stability conditions are met. A splitting rule $\underline{\alpha}$ is called *feasible* if $\underline{\alpha} \in A$.

For an arbitrary foreground job, denote by $S_i^{\underline{\alpha}}$ the sojourn time of its i-th task operating under the splitting rule $\underline{\alpha}$. This is the time it takes the i-th task to complete service at PS_i. Denote $\underline{S}^{\underline{\alpha}} = (S_1^{\underline{\alpha}}, \ldots, S_N^{\underline{\alpha}})$. The sojourn time of a foreground job through the job-split model is denoted by

$$S_0^{\underline{\alpha}} := \max\left\{S_1^{\underline{\alpha}}, \ldots, S_N^{\underline{\alpha}}\right\}. \tag{4.2}$$

Our purpose is to find a splitting rule $\underline{\alpha}^* = (\alpha_1^*, \ldots, \alpha_N^*) \in A$ such that

$$\mathbb{E}[S_0^{\underline{\alpha}^*}] = \min_{\underline{\alpha} \in A} \mathbb{E}[S_0^{\underline{\alpha}}]. \tag{4.3}$$

For a non-negative random variable X with finite (positive) first moment, the squared coefficient of variation is denoted c_X^2. Finally, heavy-traffic limits for $\xi \uparrow 1$ are taken such that the total arrival rate Λ is increased while the service-time distribution, the splitting rule $\underline{\alpha}$ and the *proportions* between the arrival rates $\lambda_0, \lambda_1, \ldots, \lambda_N$ remain fixed. Note that in this limiting regime, not all queues tend to become unstable as $\xi \uparrow 1$. More precisely, queue i becomes unstable for $\xi \uparrow 1$ only if $\xi_i = \xi$, and otherwise queue i remains stable as $\xi \uparrow 1$. Denote the set of potentially unstable queues by $U := \{i : \xi_i = \xi\}$, where ξ_i and ξ are defined in (4.1).

4.3 Analysis

In general, the "cost function" $\mathbb{E}[S_0^{\underline{\alpha}}]$ does not allow for an exact expression, and the optimization problem defined in (4.2)-(4.3) cannot be solved explicitly. The mathematical complexity is caused by the correlations between the sojourn times $S_1^{\underline{\alpha}}, \ldots, S_N^{\underline{\alpha}}$ of the jobs at the different queues. This dependence is caused by the fact that the (fragmented) foreground tasks arrive at the queues simultaneously, and by the fact that their sizes $\alpha_1 B, \ldots, \alpha_N B$ are correlated. For this reason, in this section we will develop heuristic methods to approximate $\mathbb{E}[S_0^{\underline{\alpha}}]$ and the optimal splitting rule $\underline{\alpha}^*$. The approximation of $\underline{\alpha}^*$ is based on an interpolation between two components. The first component is based on the concept of Reduced-Load Approximation (RLA), and works well in light-traffic scenarios. The second component is based on heavy-load asymptotics, and complements the RLA-based approximation for heavy-load scenarios.

In Section 4.3.1 we formulate some known results on multiclass PS models and present a number of simulation results that lead to observations that are useful for later reference. In Section 4.3.2 we outline the RLA-based approximation and in Section 4.3.3 we present the heavy-load approximation. Subsequently, in Section 4.3.4 both approximations for $\underline{\alpha}^*$ are combined into our composed-split approximation, which interpolates $\underline{\alpha}$ between these two components.

4.3.1 Preliminaries

Considering queue i in isolation, it is easy to see that this queue can be modeled as a two-class $M/G/1$-PS model, where class-1 represents the tasks originating from the foreground traffic and class-2 the background traffic. Class-1 tasks arrive according to a Poisson process with rate λ_0 and size $\alpha_i B$, where B is the size of an arbitrary job. Similarly, class-2 jobs arrive according to a Poisson process with rate λ_i and job size B. For this model, it is known that for given $B = \tau$, the conditional expected sojourn time of a foreground task of size $\alpha_i \tau$ at queue i is given by:

$$\mathbb{E}[S_i^{\underline{\alpha}} | B = \tau] = \frac{\alpha_i \tau}{1 - \xi_i}, \quad \text{and hence,} \quad \mathbb{E}[S_i^{\underline{\alpha}}] = \frac{\alpha_i \beta^{(1)}}{1 - \xi_i} \quad (i = 1, \ldots, N), \tag{4.4}$$

where $\xi_i = \beta^{(1)}(\lambda_i + \alpha_i \lambda_0)$ is the utilization of queue i. However, despite the fact that the conditional mean sojourn times $\mathbb{E}[S_i^{\underline{\alpha}} | B = \tau]$ of the *individual* tasks at each of the N queues are known, an exact expression for the mean sojourn time of *entire* foreground job, $\mathbb{E}[S_0^{\underline{\alpha}}]$ (defined in (4.2)-(4.3)), which is defined as the maximum of the (correlated) per task sojourn times is not known.

Prior to developing the approximations for the optimal splitting rule $\underline{\alpha}^*$, we perform numerical experiments based on simulations to gain insight in the optimization problem. This will lead to a number of important observations that will turn out to be useful for later reference. As an illustrative example, for the case $N = 2$ and $\beta^{(1)} = 1$, Figure 4.2 shows the behavior of $\mathbb{E}[S_0^{\underline{\alpha}}]$ for the

traffic splitting rule $\underline{\alpha} = (\alpha, 1-\alpha) \in A$ as a function of α $(0 < \alpha < 1)$, for different background and foreground load scenarios. To highlight the impact of the service-time distributions, results are shown for the extreme cases of deterministic service times (with $c_B^2 = 0$) and Pareto-2 distributed service times with $Pr(B > x) = \frac{1}{4x^2}$ for $x > 1/2$, so that $c_B^2 = \infty$.

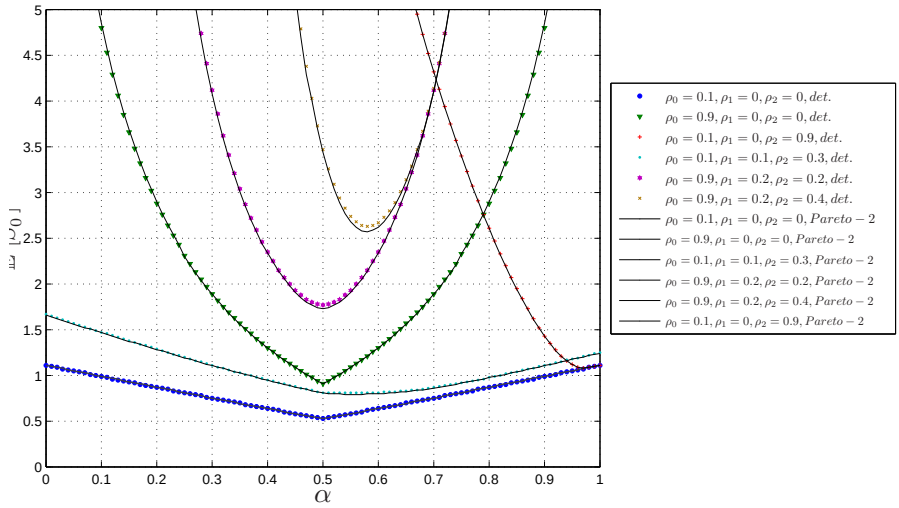


Figure 4.2: $\mathbb{E}[S_0^{\underline{\alpha}}]$ as a function of the traffic splitting ratio α, based on simulations.

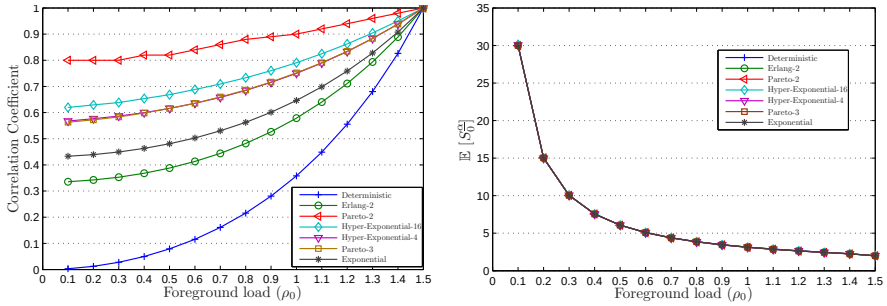
Figure 4.2 illustrates the behavioral differences between the various systems. In the absence of background traffic, the curves exhibit wedge-shaped behavior around their optimum, both for light ($\rho_0 = 0.1$) and moderate ($\rho_0 = 0.9$) foreground load. For a small background load ($\rho_1 = 0.1, \rho_2 = 0.3$) and light foreground load, the curve is nearly constant around its optimum. This is not the case if the background load becomes highly asymmetric ($\rho_1 = 0, \rho_2 = 0.9$) in the presence of light foreground load; the system is not stable for all values of α and exhibits a sharp increase in the mean sojourn time for slightly underestimated values of α (below 0.95) and is rather forgiving to overestimations where it coincides with the system without background load. Considering the outcomes depicted in Figure 4.2, it is clear that heavy foreground traffic yields a bended curve, showing that the cost function is highly sensitive to the choice α.

The most interesting observation from Figure 4.2 is that *both the cost function* $\mathbb{E}[S_0^{\underline{\alpha}}]$ *and optimal splitting ratio* $\underline{\alpha}^*$ *are nearly insensitive to the job-size distribution.* This observation is quite remarkable: Although it is well-known that the per-queue mean sojourn times are (fully) insensitive to the distribution of the job sizes (see (4.4)), no general results are known for the expected value of the maximum of the per-task sojourn times, which in general are mutually

dependent.

The question arises as to what the impact of the job-size distribution is on the correlations between the per-queue sojourn times of the foreground tasks of sizes $\alpha_1 B, \ldots, \alpha_N B$. Recall that the correlations between the sojourn times are caused by the foreground traffic, because (a) the foreground traffic stream generates simultaneous arrivals of tasks at each of the queues, and (b) the job sizes of the per-queue tasks $\alpha_1 B, \ldots, \alpha_N B$ are stochastically dependent. Intuitively, one may expect that the higher the foreground load, the stronger the correlations.

To validate this, we have performed simulation experiments for a two-queue model, with $\beta^{(1)} = 1$ and split rule $\underline{\alpha} = (1/2, 1/2)$, where ρ_0, ρ_1 and ρ_2 are parameterized as follows $\rho_1 = 1 - 2\rho_0/3$, $\rho_2 = 1/2 - \rho_0/3$, and where ρ_0 is varied between 0 and 3/2. In this way, ρ_0 is varied over the interval $[1/10, 3/2]$ such that the total load $\rho = \rho_0 + \rho_1 + \rho_2$ is kept fixed at value $\rho = 3/2$, while the ratios between ρ_1 and ρ_2 are fixed to $\rho_1 = 2\rho_2$. For each foreground job, we have calculated the statistical correlation between the two per-queue tasks (both of size $\tau/2$), and the mean sojourn times of the foreground jobs. Simulations have been run for 10^{11} jobs, which led to extremely narrow confidence intervals (not shown here). Figure 4.3a below shows the empirical correlation between the sojourn times of the foreground jobs considered as a function of ρ_0, where the service-time distributions are varied as deterministic ($c_B^2 = 0$), exponential ($c_B^2 = 1$), Erlang-2 ($c_B^2 = 1/2$), two-phase hyper-exponential with $c_B^2 = 4$ and $c_B^2 = 16$ (and balanced means), Pareto-3 (with $Pr\,(B > x) = (1 + x/2)^{-3}$ for $x > 0$ and hence $c_B^2 = 3$), and Pareto-2 (with $Pr\,(B > x) = \frac{1}{4x^2}$ for $x > 1/2$ and hence $c_B^2 = \infty$). Note that in Figure 4.3a the results for Pareto-3 and the two-phase hyper-exponential with $c_B^2 = 4$ are so close that they can hardly be distinguished. Moreover, the results in Figure 4.3b are so strikingly similar that they almost entirely overlap.



(a) Correlation as a function of ρ_0.

(b) Sojourn time as a function of ρ_0.

Figure 4.3: The correlations between the sojourn times of the foreground traffic and $\mathbb{E}[S_0^{\underline{\alpha}}]$ as a function of ρ_0 with fixed total load $\rho = 1.5$, based on simulations.

The results depicted in Figures 4.3a and 4.3b lead to the following remarkable observation.

Observation 1: *The correlations between the per-task sojourn times of the foreground traffic depend on the job-size distribution, whereas* $\mathbb{E}[S_0^{\underline{\alpha}}]$ *is nearly insensitive to the job-size distribution.*

This observation is rather intriguing. First, we observe an obvious dependence of the correlation between the per-task sojourn times with respect to the job-size distribution, where higher variability seems to imply a stronger correlation. This observation can be intuitively explained by the fact that the per-task sojourn times are positively correlated, while this correlation becomes most predominant for large job sizes, thus for outliers in the job size B. Hence, the higher the variability in the distribution of B, the more outliers in B and hence, the stronger the correlation. Second, the results show that the differences in correlations over the different service-time distributions do not manifest themselves in significant differences in $\mathbb{E}[S_0^{\underline{\alpha}}]$. The impact of the correlations between the per-task sojourn times "vanishes" when looking at $\mathbb{E}[S_0^{\underline{\alpha}}]$. This observation will turn out to be useful for developing an approximation for $\mathbb{E}[S_0^{\underline{\alpha}}]$, see Sections 4.3.3 and 4.3.4.

4.3.2 Reduced Load Approximation (RLA)

The Reduced Load Approximation (RLA) splits jobs into tasks according to the split rule presented in Section 3.2:

$$\alpha^*_{RLA,i} := \frac{1-\rho_i}{\sum_{j=1}^{N}(1-\rho_j)} \quad (i = 1, \ldots, N). \tag{4.5}$$

This simple splitting rule $\underline{\alpha}^*_{RLA} = (\alpha^*_{RLA,1}, \ldots, \alpha^*_{RLA,N})$ cuts the foreground jobs into i tasks, each with a size proportional to $1-\rho_i$, i.e., the average amount of capacity not used by the background traffic at queue i. Note that the RLA is insensitive to the job-size distribution (except of course for its mean $\beta^{(1)}$), which is in line with Observation 1 in Section 4.3.1. In addition to its attractive simplicity, the RLA (4.5) is asymptotically tail optimal for the case of regularly varying service-time distributions (see Chapter 3 for details).

Extensive numerical experimentation in Chapter 3 (and Table 4.1 in Section 4.4 below) reveals that the RLA leads to highly accurate approximations of $\underline{\alpha}^*$ if either $\rho_0 \approx 0$ or if the queues are fairly equally loaded, but that it may become inaccurate when one or more queues are heavily loaded while others are not (see for example the right upper corner in Table 4.1 below). This raises the need for a refinement of the RLA to improve its accuracy for asymmetric background-load scenarios, which is the main goal of the present chapter. To this end, in the next section we use and combine known heavy-traffic asymptotics for multi-class PS models and Observation 1 to derive approximations for $E[S_0^{\underline{\alpha}}]$, and hence for the optimal splitting of jobs $\underline{\alpha}^*$, under heavy-traffic assumptions.

4.3.3 Heavy Traffic Approximation (HTA)

In this section we will use heavy-traffic (HT) asymptotics for multi-class PS models to develop an approximation for $\mathbb{E}[S_0^{\underline{\alpha}}]$, and hence of $\underline{\alpha}^*$, that meets these asymptotic HT-properties. To formulate these HT properties, recall that queue i considered in isolation can be modeled as a two-class $M/G/1$-PS model, where class-1 jobs (representing background jobs at queue i) arrive according to a Poisson process with rate λ_0 and service times $\alpha_i B$, and where class-2 jobs (representing foreground jobs) arrive according to a Poisson process with rate λ_i and service times B. Let $S_i(\tau)$ denote the sojourn time of an arbitrary job of size τ at queue i (regardless of its class). Zwart and Boxma [128] show that for $\tau > 0$, $i \in U$, $\underline{\alpha} \in A$,

$$(1-\xi)S_i(\alpha_i\tau) \to_d \Theta(\alpha_i\tau) \quad (\xi \uparrow 1), \tag{4.6}$$

where $\Theta(\zeta)$ is an exponentially distributed random variable with mean ζ. Moreover, in [128] it is shown also that moment-wise convergence holds: For $\tau > 0$, $i \in U$, $\underline{\alpha} \in A$ and $k = 1, 2, \ldots$,

$$\lim_{\xi\uparrow 1} (1-\xi)^k \mathbb{E}[S_i(\alpha_i\tau)^k] = k!\alpha_i^k\tau^k. \tag{4.7}$$

To this end, it is important to observe that the conditional results in (4.6)-(4.7) were proven for the classical single-class $M/G/1$-PS queue, but are also directly applicable to multi-class $M/G/1$-PS queues. Then, removing the conditioning with respect to the distribution of the size of the foreground tasks at queue i in (6) leads to the following result for S_i, the unconditional sojourn time for a job at queue i: For $\tau > 0$, $i \in U$, $\underline{\alpha} \in A$,

$$(1-\xi)S_i \to_d \Theta(\alpha_i)B \quad (\xi \uparrow 1), \tag{4.8}$$

where $\Theta(\alpha_i)$ is exponentially distributed with mean α_i, B is the job size, and where the random variables $\Theta(\alpha_i)$ and B are independent. Moreover, the k-th moment of the unconditioned sojourn time of an arbitrary foreground job at queue i has the following limiting HT-behavior: For $i \in U$, $\underline{\alpha} \in A$, $k = 1, 2, \ldots$,

$$\lim_{\xi\uparrow 1} (1-\xi)^k \mathbb{E}[S_i^k] = k!\alpha_i^k\beta^{(k)}. \tag{4.9}$$

We are now ready to use the HT-asymptotics (4.6)-(4.9) to develop a simple approximation for $\mathbb{E}[S_0^{\underline{\alpha}}]$ that works well in HT-conditions, i.e. where $\xi \uparrow 1$. In the absence of an exact analysis, we will construct a simple approximation for the joint probability distribution of

$$\underline{S}_{HTA}^{\underline{\alpha}}(\tau) = (S_{HTA,1}^{\underline{\alpha}}(\alpha_1\tau), \ldots, S_{HTA,N}^{\underline{\alpha}}(\alpha_N\tau)), \tag{4.10}$$

where $S_{HTA,i}^{\underline{\alpha}}(\alpha_i\tau)$ is the sojourn time of the i-th fragment of an arbitrary foreground job of size τ (note that this fragment itself is of size $\alpha_i\tau$). To this end, note that (4.6) implies the following limiting behavior for the *marginal*

distributions of the conditional sojourn times $S^{\underline{\alpha}}_{HTA,i}(\alpha_1\tau)$: For $i \in U$, $\underline{\alpha} \in A$, $t > 0$,

$$\lim_{\xi\uparrow 1} Pr\{(1-\xi)S_{HTA,i}(\alpha_i\tau) > t\} = \exp\left\{-\frac{t}{\alpha_i\tau}\right\}. \tag{4.11}$$

Note that (4.11) is only valid for $i \in U$, because if $i \notin U$ then queue i will not become unstable so that $(1-\xi)S_{HTA,i}(\alpha_i\tau) \to 0$ (a.s.), when $\xi \uparrow 1$ (see also Remark 4.3.2).

Next, we develop an approximation for $\mathbb{E}[S^{\underline{\alpha}}_0]$ which satisfies the known heavy-traffic properties of the marginal per-task sojourn-time distributions formulated in (4.10)-(4.11). Regarding the correlations, recall from Observation 1 (formulated in Section 4.3) that $\mathbb{E}[S^{\underline{\alpha}}_0]$ at best weakly depends on the correlations between the per-task sojourn times. Therefore, although the per-task conditional sojourn times $S^{\underline{\alpha}}_{HTA,1}(\alpha_1\tau), \ldots, S^{\underline{\alpha}}_{HTA,N}(\alpha_N\tau)$ are clearly not independent, we *assume* that they are. Based on this assumption, we *approximate* the distribution of $\underline{S^{\underline{\alpha}}_{HTA}}(\tau)$ (defined in (4.10)) by assuming that $S^{\underline{\alpha}}_{HTA,1}(\alpha_1\tau), \ldots, S^{\underline{\alpha}}_{HTA,N}(\alpha_N\tau)$ (a) are exponentially distributed with mean $\alpha_i\tau/(1-\xi_i)$, so that the marginal distributions satisfy the known HT behavior in (4.11) for $i \in U$ (see also Remark 4.3.2), and (b) are mutually independent:

$$Pr\{S^{\underline{\alpha}}_{HTA,1}(\alpha_1\tau) > t_1, \ldots, S^{\underline{\alpha}}_{HTA,N}(\alpha_N\tau) > t_N\} \approx \prod_{i=1}^{N} \exp\left\{-\frac{1-\xi_i}{\alpha_i\tau}t_i\right\}. \tag{4.12}$$

We will now use the approximation in (4.12), which covers the known HT-limiting behavior from (4.6)-(4.7), to derive a simple approximation for $E[S^{\underline{\alpha}}_0]$ that works well when $\xi \uparrow 1$. To this end, for $\tau > 0$, $\underline{\alpha} \in A$, define

$$S^{\underline{\alpha}}_{HTA,0}(\tau) := \max\{S^{\underline{\alpha}}_{HTA,1}(\alpha_1\tau), \ldots, S^{\underline{\alpha}}_{HTA,N}(\alpha_N\tau)\}. \tag{4.13}$$

Then using (4.12), the distribution of $S^{\underline{\alpha}}_{HTA,0}(\tau)$ can be approximated as the distribution of the maximum of N independent exponentially distributed random variables with known parameters (given in (4.12)). See Property 1 in Appendix A for an expression for the mean value of such random variable. Using this result, the conditional cost function $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}(\tau)]$ can be readily obtained from Property 1 simply by making following substitutions in (A.1) from Appendix A: For $i = 1, \ldots, N$,

$$X_i := S^{\underline{\alpha}}_{HTA,i}(\alpha_i\tau), \quad \text{and} \quad 1/\mu_i := \mathbb{E}[S^{\underline{\alpha}}_{HTA,i}(\alpha_i\tau)] = \frac{\alpha_i\tau}{1-\xi_i}. \tag{4.14}$$

Note that it is readily verified that the so-obtained approximation of $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}(\tau)]$ is *linear* in τ, so that the unconditioned cost function $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$ can be directly obtained by replacing τ by $\beta^{(1)}$ in (4.14). In this way, the cost function

$\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$ can be approximated by using (4.12)-(4.14) and (A.1), which leads to the following approximation for $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$:

$$\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}] \approx \beta^{(1)} \sum_{k=1}^{N} (-1)^{k+1} \sum_{(i_1,\ldots,i_k)\in S_k} \frac{1}{\frac{1-\xi_{i_1}}{\alpha_{i_1}} + \cdots + \frac{1-\xi_{i_N}}{\alpha_{i_N}}}, \tag{4.15}$$

where

$$S_k := \{(i_1,\ldots,i_k) : i_1,\ldots,i_k \in \{1,\ldots,N\}, i_1 < i_2 < \cdots < i_k\}. \tag{4.16}$$

Note that the approximate expression for $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$ in (4.15) is explicit, and hence, the computation time is negligible. Notice also that the right-hand side of (4.15) is (fully) insensitive to the job-size distribution, which is in line with Observation 1 discussed in Section 4.3.1.

Next, denote by $\underline{\alpha}^*_{HTA} = (\alpha^*_{HTA,1},\ldots,\alpha^*_{HTA,N})$ the splitting rule that minimizes $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$ among all $\underline{\alpha} \in A$, i.e.,

$$\mathbb{E}[S^{\underline{\alpha}^*}_{HTA,0}] = \min_{\underline{\alpha}\in A} \mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]. \tag{4.17}$$

The optimal split $\underline{\alpha}^*_{HTA}$ can then be approximated by evaluating (4.15) over all $\underline{\alpha} \in A$, or by some non-linear optimization method. In practice, this causes no problem as N is not too large. Note that in the context of concurrent access for wireless networks, which was the main motivation for this study, N is indeed small, say 2 or 3. We refer to Remark 4.4.2 for a brief discussion on the complexity of the optimization.

4.3.4 Composed Split Approximation (CSA)

Now we combine both approaches in the following to obtain the composed-split approximation (CSA), denoted $\underline{\alpha}^*_{CSA} = (\alpha^*_{CSA,1},\ldots,\alpha^*_{CSA,N})$, where for $i = 1,\ldots,N$,

$$\alpha^*_{CSA,i} = (1-\kappa_i)\alpha^*_{RLA,i} + \kappa_i \alpha^*_{HTA,i}, \quad \text{with } \kappa_i := \max\{\rho_1,\ldots,\rho_N\}, \tag{4.18}$$

and where $\alpha^*_{RLA,i}$ is given in (4.5), and where $\alpha^*_{HTA,i}$ can be obtained from (4.15)-(4.17). The interpolation factor κ_i is taken such that κ_i is independent of $\underline{\alpha}$ (and of i, see also Remark 4.3.3), while for light-traffic scenarios the RLA is dominant and for heavy-load scenarios the HTA is dominant. We refer Remark 4.3.3 below for a discussion on the choice of the interpolation factor.

Remark 4.3.1 (RLA optimizing the lower bound)**.** A complicating factor of the model is the complex correlation structure between the sojourn times of the individual tasks after a job has been split. In this context, we observe that the reduced load approximation (RLA) can be thought of as implying that there is "perfect" correlation in the sojourn times of the different tasks, and thus gives a

lower bound on the mean sojourn times. To this end, note that under "perfect correlation", for all $i, j = 1, \ldots, N$, $\underline{\alpha} \in A$ and $\tau > 0$,

$$\mathbb{E}[S_i^{\underline{\alpha}}|B = \tau] = \mathbb{E}[S_j^{\underline{\alpha}}|B = \tau]. \tag{4.19}$$

Using (4.4), this set of equations (4.19) is readily seen to lead to the RLA defined in (4.5). In this way, the RLA can be seen as optimizing a lower bound for $\mathbb{E}[S_0^{\underline{\alpha}}]$.

On the contrary, the HTA can be viewed as optimizing an upper bound for $\mathbb{E}[S_0^{\underline{\alpha}}]$. To this end, note that the simulation results shown in Figure 4.3 suggest that for fixed $\underline{\alpha} \in A$ the per-task sojourn times are positively correlated. Loosely speaking, the maximum of positively correlated random variables is stochastically dominated by the maximum of independent random variables with identical marginal distributions. Hence, such a dominance also holds for the expected values, so that the HTA defined in (4.15) gives an upper bound for $\mathbb{E}[S_0^{\underline{\alpha}}]$, defined in (4.3). In that sense, the HTA is optimizing an upper bound.

Remark 4.3.2 (HT behavior)**.** Recall that the HT behavior in (4.11) holds if and only if $i \in U$, i.e. for those queues i for which $\xi_i = \xi$, defined in (4.1). For $i \in U$, (4.11) shows that the conditional sojourn time for queue i converges (both in distribution and moment-wise) to an exponential distribution with known mean. In contrast, the queues $i \notin U$ do not become unstable for $\xi \uparrow 1$. However, note that in the HTA in (4.15) the marginal conditional sojourn-time distributions are approximated by (independent) exponential distributions for all $i = 1, \ldots, N$. Nonetheless, note that in HT-conditions (i.e., $\xi \approx 1$) the impact of the per-task sojourn times of queues i for $i \notin U$ on $\mathbb{E}[S_0^{\underline{\alpha}}]$ tends to vanish under HT-scalings.

Remark 4.3.3 (Interpolation factor κ_i)**.** The choice of the interpolation factor κ_i in (4.18) only depends on the background-load values. The benefit is its simplicity and the fact that κ_i does not depend on $\underline{\alpha}$, the parameter which is to be optimized. The drawback of this choice is that it does not accurately cover the HT behavior when $\xi \uparrow 1$ while $\rho_1, \ldots, \rho_N$ are close to 0; this may happen when the foreground load ρ_0 is large. One way to overcome this problem is to take as the interpolating factor $\kappa_i := \xi_i$, defined in (4.1). The problem is that in this way, the interpolation factor itself depends on α_i which leads to a fixed-point equation to solve for α_i. We have checked the accuracy of the approximations based on $\kappa_i = \xi_i$; note that convergence of such fixed-point iteration can easily be shown to hold. Our results show that no significant improvement of the accuracy of the approximations is obtained.

4.4 Numerical results

To assess the accuracy of the approximations for the optimal splitting rule $\underline{\alpha}$, we have performed extensive numerical experimentation, comparing the approximation results with simulations. To cover a wide range of parameter combinations

in a systematic manner, we have varied the job-size distributions (deterministic, exponential, hyper-exponential, Pareto), and the load values of the foreground traffic (high, medium, low) and the background traffic (high, medium, low).

In our experiments various parameters scenarios were considered. For these parameter combinations, we calculated the following:

1. the optimal split rule $\underline{\alpha}^* = (\alpha_1^*, \ldots, \alpha_{N-1}^*, 1 - \alpha_1^* - \cdots - \alpha_{N-1}^*)$,
2. approximations for $\underline{\alpha}^*$, denoted $\underline{\alpha}_{app}^* = (\alpha_{app,1}^*, \ldots, \alpha_{app,N-1}^*, 1 - \alpha_{app,1}^* - \cdots - \alpha_{app,N-1}^*)$, and
3. the relative difference in the mean foreground traffic processing times, defined by

$$\Delta\% := abs\left(\frac{\mathbb{E}[S_0^{\underline{\alpha}_{app}^*}] - \mathbb{E}[S_0^{\underline{\alpha}^*}]}{\mathbb{E}[S_0^{\underline{\alpha}^*}]}\right) \times 100\%. \tag{4.20}$$

First, we assume $N = 2$ and $\beta^{(1)} = 1$. The job-size distributions were varied as deterministic (to cover the case $c_B^2 = 0$), exponential ($c_B^2 = 1$), H_2 with $c_B^2 = 16$ (with balanced means) and Pareto-2 (with $c_B^2 = \infty$). The load of the foreground traffic ρ_0 was varied as 0.1, 0.5, 0.9 and 1.8, and the background loads ρ_1 and ρ_2 were varied as $0.1, 0.3, \ldots, 0.9$. To search for the optimal splitting rule $\underline{\alpha}^* = (\alpha, 1 - \alpha)$, we evaluated all feasible values of α with a step size 0.01, and more finely if needed. Below we will present the results of the evaluations. Tables 4.1 to 4.5 show for each feasible combination of ρ_1 and ρ_2 the corresponding values of the optimal split determined by simulation $\underline{\alpha}^* = (\alpha^*, 1 - \alpha^*)$, the approximated optimal split $\underline{\alpha}_{app}^* = (\alpha_{app}^*, 1 - \alpha_{app}^*)$ for $app \in \{\text{RLA}, \text{HTA}, \text{CSA}\}$, and the relative error in the cost function $\Delta\%$, defined in (4.20). To obtain highly accurate simulation results, experiments were run with extremely many jobs, up to 10^{10} if needed, leading to very narrow confidence intervals (CIs), such that all digits in the Table 4.1 to 4.5 below are significant. For compactness of the presentation the CIs are not shown. Also, note that because of the symmetry in the model for $N = 2$ only the results for $\rho_1 \leq \rho_2$ are shown.

To start, we compare the performance of the RLA (discussed in Section 4.3.2), the HTA (discussed in Section 4.3.3) and the CSA (discussed in Section 4.3.4). As an illustrative example, Table 4.1 shows the results for the case with $\rho_0 = 0.1$ and exponential job-size distributions, for a variety of background-load combinations of (ρ_1, ρ_2).

We observe that the CSA indeed performs much better than both the RLA and the HTA. In fact, the RLA performs very well if $\rho_1 \approx \rho_2$, but tends to degrade significantly if ρ_1 and ρ_2 are far apart. This degradation in the accuracy becomes most apparent if $\rho_1 \approx 0$ and $\rho_2 \approx 1$, showing double-digit error percentages. This was to be expected, since the RLA (4.5) simply splits traffic

Reduced Load Approximation (RLA)

	α^*	α^*_{RLA}	$\Delta\%$	α^*	α^*_{RLA}	$\Delta\%$	α^*	α^*_{RLA}	$\Delta\%$	α^*	α^*_{RLA}	$\Delta\%$	α^*	α^*_{RLA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.56	0.03	0.70	0.64	1.92	0.85	0.75	8.79	0.97	0.90	24.88
0.3				0.50	0.50	0.00	0.63	0.58	0.69	0.80	0.70	6.19	0.96	0.88	23.39
0.5							0.50	0.50	0.00	0.70	0.63	2.58	0.94	0.83	19.65
0.7										0.50	0.50	0.00	0.86	0.75	10.76
0.9													0.50	0.50	0.00

Heavy-traffic approximation (HTA)

	α^*	α^*_{HTA}	$\Delta\%$	α^*	α^*_{HTA}	$\Delta\%$	α^*	α^*_{HTA}	$\Delta\%$	α^*	α^*_{HTA}	$\Delta\%$	α^*	α^*_{HTA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.63	2.29	0.70	0.78	2.79	0.85	0.91	3.01	0.97	0.99	1.71
0.3				0.50	0.50	0.00	0.63	0.67	0.66	0.80	0.85	1.50	0.96	0.98	1.54
0.5							0.50	0.50	0.00	0.70	0.73	0.32	0.94	0.96	0.93
0.7										0.50	0.50	0.00	0.86	0.87	0.12
0.9													0.50	0.50	0.00

Composed-split approximation (CSA)

	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.59	0.42	0.70	0.72	0.07	0.85	0.87	0.15	0.97	0.98	0.72
0.3				0.50	0.50	0.00	0.63	0.63	0.00	0.80	0.81	0.03	0.96	0.97	0.50
0.5							0.50	0.50	0.00	0.70	0.70	0.01	0.94	0.95	0.22
0.7										0.50	0.50	0.00	0.86	0.86	0.01
0.9													0.50	0.50	0.00

Table 4.1: Comparison of the RLA, the HTA and the CSA for the case of exponential job-size distributions and $\rho_0 = 0.1$, and $N = 2$.

proportional to the *relative* amounts of capacity not used by the background traffic, while one may suspect that the *absolute* values of the background traffic have a large impact on the sensitivity of the choice of the splitting rule with respect to the background-load values. As expected, the HTA is doing much better in those asymmetric heavy-load scenarios, but tends to degrade somewhat when PS_1 is lightly loaded ($\rho_1 = 0.1$) and PS_2 is moderately loaded ($\rho_2 = 0.7$), leading to errors up to 3%. We observe that the CSA overall performs significantly better than the RLA and the HTA, in most cases leading to an error percentage less than 0.5%.

In short, the results in Table 4.1 indeed show that the usefulness of refining the simple and explicit RLA (introduced in Chapter 3) into the CSA, leading to extremely accurate results for most of the parameter combinations.

In the remainder of this section, we will further evaluate the accuracy of the CSA for a broad range of service-time distributions and combinations of foreground and background loads. Table 4.2 shows the results for light foreground load, with $\rho_0 = 0.1$. The results show that the approximations are highly accurate over a wide range of background-load combinations and service-time distributions, with errors significantly less than 1%. The least favorable results from our approximation were consistently found when the background load is highly asymmetric ($\rho_1 = 0.1$ and $\rho_2 = 0.9$), but even in those cases the results are highly accurate, with errors below 0.8%.

deterministic															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.59	0.43	0.71	0.72	0.03	0.86	0.87	0.10	0.97	0.98	0.70
0.3				0.50	0.50	0.00	0.63	0.63	0.01	0.80	0.81	0.03	0.96	0.97	0.49
0.5							0.50	0.50	0.00	0.71	0.70	0.00	0.94	0.95	0.05
0.7										0.50	0.50	0.00	0.86	0.86	0.03
0.9													0.50	0.50	0.00
exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.59	0.42	0.70	0.72	0.07	0.85	0.87	0.15	0.97	0.98	0.72
0.3				0.50	0.50	0.00	0.63	0.63	0.00	0.80	0.81	0.03	0.96	0.97	0.50
0.5							0.50	0.50	0.00	0.70	0.70	0.01	0.94	0.95	0.22
0.7										0.50	0.50	0.00	0.86	0.86	0.01
0.9													0.50	0.50	0.00
hyper-exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.59	0.76	0.70	0.72	0.22	0.85	0.87	0.12	0.97	0.98	0.76
0.3				0.50	0.50	0.00	0.63	0.63	0.07	0.80	0.81	0.04	0.96	0.97	0.57
0.5							0.50	0.50	0.00	0.70	0.70	0.02	0.94	0.95	0.23
0.7										0.50	0.50	0.00	0.86	0.86	0.13
0.9													0.50	0.50	0.00
Pareto															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.55	0.59	0.52	0.70	0.72	0.15	0.85	0.87	0.23	0.97	0.98	0.73
0.3				0.50	0.50	0.00	0.62	0.63	0.03	0.79	0.81	0.01	0.96	0.97	0.57
0.5							0.50	0.50	0.00	0.70	0.70	0.01	0.94	0.95	0.25
0.7										0.50	0.50	0.00	0.86	0.86	0.03
0.9													0.50	0.50	0.00

Table 4.2: Simulation results for light foreground load ($\rho_0 = 0.1$), $N = 2$.

deterministic															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.58	0.11	0.68	0.68	0.03	0.81	0.81	0.02	0.95	0.96	0.25
0.3				0.50	0.50	0.00	0.61	0.61	0.03	0.75	0.75	0.04	0.92	0.92	0.05
0.5							0.50	0.50	0.00	0.65	0.65	0.02	0.85	0.85	0.20
0.7										0.50	0.50	0.00			
exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.58	0.13	0.68	0.68	0.04	0.81	0.81	0.01	0.95	0.96	0.28
0.3				0.50	0.50	0.00	0.61	0.61	0.03	0.75	0.75	0.06	0.92	0.92	0.06
0.5							0.50	0.50	0.00	0.65	0.65	0.04	0.85	0.85	0.22
0.7										0.50	0.50	0.00			
hyper-exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.58	0.35	0.68	0.68	0.16	0.81	0.81	0.05	0.95	0.96	0.48
0.3				0.50	0.50	0.00	0.61	0.61	0.00	0.75	0.75	0.07	0.92	0.92	0.15
0.5							0.50	0.50	0.00	0.65	0.65	0.09	0.85	0.85	0.36
0.7										0.50	0.50	0.00			
Pareto															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.58	0.19	0.67	0.68	0.05	0.81	0.81	0.04	0.95	0.96	0.22
0.3				0.50	0.50	0.00	0.60	0.61	0.01	0.74	0.75	0.00	0.92	0.92	0.18
0.5							0.50	0.50	0.00	0.65	0.65	0.07	0.85	0.85	0.06
0.7										0.50	0.50	0.00			

Table 4.3: Simulation results for moderate foreground load ($\rho_0 = 0.5$), $N = 2$.

Tables 4.3 and 4.4 show the results for moderate foreground load values of $\rho_0 = 0.5$ and $\rho_0 = 0.9$, respectively. The results again show that the CSA performs extremely well in all cases considered, with errors significantly less than 1%, even for Pareto-2 distributed job sizes (thus with infinite variance) and highly asymmetric background-load scenarios. Note that in Tables 4.3 and 4.4 several combinations of (ρ_1, ρ_2) are omitted because they violate the stability conditions.

Finally, to assess the accuracy of the approximation for heavily loaded foreground traffic, additional simulation runs were conducted for $\rho_0 = 1.8$. Note that in this case, the set of α-values for which the system is still stable is limited to $\alpha \in (4/9; 1/2)$. To obtain an accurate estimate for α^*, we simulated $\mathbb{E}[S_0^{(\alpha,1-\alpha)}]$ for different α-values with step size 0.001.

The results shown in Table 4.5 demonstrate that the CSA is also extremely accurate for heavy foreground load scenarios. Note that the mean sojourn times and the optimal split are indeed remarkably insensitive with respect to the job-size distributions, which supports the validity of Observation 1 in Section 4.3.

To assess the accuracy of the approximation for $N > 2$, we also consider a model with $N = 3$ with Pareto-2 distributed service times with $\beta^{(1)} = 1$. Ta-

deterministic															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.57	0.02	0.66	0.66	0.02	0.77	0.77	0.06	0.91	0.91	0.63
0.3				0.50	0.50	0.00	0.59	0.59	0.02	0.71	0.71	0.38			
0.5							0.50	0.50	0.00						
exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.57	0.07	0.66	0.66	0.04	0.77	0.77	0.17	0.91	0.91	0.11
0.3				0.50	0.50	0.00	0.59	0.59	0.05	0.71	0.71	0.06			
0.5							0.50	0.50	0.00						
hyper-exponential															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.57	0.11	0.66	0.66	0.09	0.77	0.77	0.12	0.91	0.91	0.07
0.3				0.50	0.50	0.00	0.59	0.59	0.01	0.71	0.71	0.36			
0.5							0.50	0.50	0.00						
Pareto															
	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$	α^*	α^*_{CSA}	$\Delta\%$
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.50	0.50	0.00	0.57	0.57	0.08	0.66	0.66	0.05	0.77	0.77	0.00	0.91	0.91	0.00
0.3				0.50	0.50	0.00	0.59	0.59	0.00	0.71	0.71	0.53			
0.5							0.50	0.50	0.00						

Table 4.4: Simulation results for moderate foreground load ($\rho_0 = 0.9$), $N = 2$.

	α^*	α^*_{CSA}	$\Delta\%$
deterministic	0.473	0.47312	0.03
exponential	0.473	0.47312	0.06
hyper-exponential	0.474	0.47312	0.05
Pareto	0.474	0.47312	0.06

Table 4.5: Simulation results for heavy foreground load ($\rho_0 = 1.8$), with $N = 2$, $\rho_1 = 0.1$ and $\rho_2 = 0.0$.

ble 4.6 shows the results for this system with $\rho_0 = 1.5$ where ρ_1, ρ_2 and ρ_3 are varied as 0.3, 0.5, and 0.7. For each feasible parameter setting, the table shows $\underline{\alpha}^* = (\alpha_1^*, \alpha_2^*, \alpha_3^*)^\top$ obtained via simulation, $\underline{\alpha}^*_{CSA} = (\alpha_1{}^*_{CSA}, \alpha_2{}^*_{CSA}, \alpha_3{}^*_{CSA})^\top$, and the relative error defined in (4.20). The results in Table 4.6 show that the

$\rho_3 = 0.3$															
	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.366 0.366 0.268	0.362 0.362 0.276	0.39%	0.399 0.301 0.300	0.394 0.303 0.303	0.13%	0.437 0.230 0.333	0.435 0.233 0.332	0.42%	0.484 0.144 0.372	0.482 0.150 0.368	0.89%	0.536 0.051 0.413	0.535 0.053 0.412	0.47%
0.3				0.333 0.333 0.334	0.334 0.333 0.333	0.00%	0.370 0.260 0.370	0.370 0.261 0.369	0.27%	0.415 0.170 0.415	0.414 0.172 0.414	0.34%		unstable	
0.5							0.292 0.292 0.416	0.293 0.293 0.414	0.24%		unstable			unstable	
$\rho_3 = 0.5$															
	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.400 0.400 0.200	0.396 0.395 0.209	1.03%	0.437 0.333 0.230	0.435 0.332 0.233	0.42%	0.485 0.257 0.258	0.480 0.260 0.260	0.40%	0.538 0.170 0.292	0.535 0.172 0.293	0.50%		unstable	
0.3				0.370 0.370 0.260	0.370 0.369 0.261	0.27%	0.416 0.292 0.292	0.414 0.293 0.293	0.24%		unstable			unstable	
$\rho_3 = 0.7$															
	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%	$\underline{\alpha}^*$	$\underline{\alpha}^*_{CSA}$	Δ%
ρ_1 \ ρ_2	0.1			0.3			0.5			0.7			0.9		
0.1	0.440 0.440 0.120	0.435 0.434 0.131	1.60%	0.484 0.372 0.144	0.482 0.368 0.150	0.89%	0.538 0.292 0.170	0.535 0.293 0.172	0.50%		unstable			unstable	
0.3				0.415 0.415 0.170	0.414 0.414 0.172	0.34%		unstable			unstable			unstable	

Table 4.6: Simulation results for moderate foreground load ($\rho_0 = 1.5$), $N = 3$.

accuracy of the CSA is again excellent for $N = 3$. To summarize, the results in Tables 4.1 to 4.6 show that the CSA, which combines the benefits of the RLA and the HTA, leads to extremely accurate approximations for $\underline{\alpha}^*$ over a wide range parameter settings.

We end this section with a number of remarks.

Remark 4.4.1 (Insensitivity)**.** In Section 4.3.1 we observed on the basis of preliminary simulation experiments that the mean sojourn times, and the optimal $\underline{\alpha}$-values, are remarkably insensitive with respect to the service-time distributions (Observation 1), even for extremely variable job-size distributions. In this context, notice that the results shown in Tables 4.1 to 4.5 confirm this observation.

Remark 4.4.2 (Larger values for N)**.** Despite the fact the RLA for $\underline{\alpha}^*$ in (4.5) is explicit, the HTA is not, and can only be calculated numerically. This generally requires numerical optimization of $\underline{\alpha}$ over the set A. When N is not too large this causes no problem, because the evaluation of $\mathbb{E}[S^{\underline{\alpha}}_{HTA,0}]$ for given $\underline{\alpha}$ is explicit, and the set of $\underline{\alpha}$-values is within the bounded set A for which standard non-linear optimization techniques are available. Note that discretization of the set A and

then enumeration over all $\underline{\alpha}$ can also be done for small N. However, when N is large, the computation times may become significant. In those cases, a further simplification of the HTA seems to be needed. In this context, recall that in general not all queues become unstable as $\xi \uparrow 1$ (namely only those for which $i \in U$, see also Remark 4.3.2). Therefore, only the proper choice of α_i for $i \in U$ may be crucial, whereas the cost function $\mathbb{E}[S_0^{\underline{\alpha}}]$ may be expected to be fairly insensitive to the choice of α_i for $i \notin U$ as $\xi \uparrow 1$, i.e., becomes negligible in heavy traffic. This observation may lead to a dramatic reduction of the dimension of the optimization problem. Furthermore, for $N \to \infty$ one may use asymptotic results from the powerful extreme-value theory to develop approximations for the HT behavior of $\mathbb{E}[S_0^{\underline{\alpha}}]$. These observations open up possibilities for further reducing the computational complexity of the calculation of $\underline{\alpha}^*_{HTA}$. Finally, note that from an application point of view, in the context of wireless networks with concurrent access (a) N is rather small, say $N \leq 3$, and (b) the job-split ratio does not have to be (re)calculated in real time, so that the computational requirements are not very strict.

Remark 4.4.3 (Accuracy)**.** Each approximation method, almost by definition, has parameter combinations for which the results become inaccurate. As for the CSA, the numerical results in Tables 4.1 to 4.5 suggest that the CSA performs extremely well and is resilient to high and asymmetric background loads and extremely variable service-time distributions. We suspect that the CSA can be "blown up" and become less accurate in two extreme cases, where the cost function $\mathbb{E}[S_0^{\underline{\alpha}}]$ becomes extremely sensitive to the choice of the system parameters. First, a potential source of inaccuracy is when $\rho_0 \approx 2$ (for the case $N = 2$). To this end, note that the HTA presented in Section 4.3.2 explicitly uses the known heavy-traffic result for the $M/G/1$-PS queue, stating that the marginal conditional sojourn times (properly scaled) converge to an exponential distribution with known mean [128], and then simply assumes that the individual per-task sojourn times are mutually independent. However, it should be noted that this independence assumption may be unrealistic, particularly when the foreground load is high and the background traffic is negligible. Most remarkably, the results in Table 4.5 show that despite the fact that our HT-approximation for $\mathbb{E}[S_0^{\underline{\alpha}}]$ (which assumes independence) completely neglects the dependence between the sojourn times, the corresponding minimum $\underline{\alpha}^*_{CSA}$ is strikingly close to the optimum. We suspect that the CSA can be "blown up" by putting the correlation of the per-task sojourn times to an extreme, by taking $\rho_1 = \epsilon$, $\rho_2 = 0$ and $\rho_0 = 2 - 2\epsilon$ for some $\epsilon \approx 0$. In such cases, the cost function $\mathbb{E}[S_0^{\underline{\alpha}}]$ is extremely sensitive to the choice of $\underline{\alpha}$. On the contrary, we also notice that in that case the set of possible values of α for which both queues are stable shrinks to

$$\alpha \in \left(\frac{1}{2} - \frac{\epsilon}{2 - 2\epsilon}; \frac{1}{2} \right), \tag{4.21}$$

so that $\alpha \to 1/2$ as $\epsilon \to 0$. A second potential source of inaccuracy is when the asymmetry in the background load is put to an extreme, say $\rho_1 = 1 - \epsilon$ and $\rho_2 = 0$, for $\epsilon \approx 0$ (for the case $N = 2$). In those hypothetical cases, the

cost function also becomes extremely sensitive to any additional load added to queue 1, so that even the smallest deviation form the optimal split may lead to significant degradation of the accuracy of the approximation.

4.5 Conclusions and further research

The results presented in this chapter raise a number of challenging topics for further research. First, the simulation results in Figures 4.2 and 4.3 show that $\mathbb{E}[S_0^{\alpha}]$ is at least near-insensitive with respect to the distribution of the job size B. The question arises whether $\mathbb{E}[S_0^{\alpha}]$ is fully insensitive to the distribution of B, similar to the marginal distributions of the per-task sojourn times (see (4.4)). Even extremely long simulation runs do not give a definite answer whether $\mathbb{E}[S_0^{\alpha}]$ is fully insensitive to the distribution of B. Obtaining rigorous proofs of insensitivity properties, possibly under additional requirements on the job-size distribution, is a challenging topic for further research. Second, an intriguing observation made in Figures 4.3a and 4.3b is that the differences in correlations over different job-size distributions are evident but seem to have no impact on $\mathbb{E}[S_0^{\alpha}]$. The impact of the correlations between the per-task sojourn times seems to cancel out when evaluating $\mathbb{E}[S_0^{\alpha}]$. Currently, a full understanding of this phenomenon is lacking, and is left as a topic for follow-up research. Third, the Poisson assumption may be relaxed. In fact, we suspect that both the RLA and the HTA can be quite easily extended for example to renewal arrival processes, but it is unclear to what extent they can be further generalized to more realistic arrival processes that include correlations between job arrivals. Derivation of such approximations is a challenging subject for further research. Finally, in our model the PS-queues are assumed to have unit capacity. In further research, the job-split model may be extended to also incorporate the capacity of the queues.

Chapter 5

Dynamic Traffic Assignment to Multiple Networks

This chapter considers a networking environment in which specific users are able to select between the available wireless networks with the aim to minimize the expected time to download files in the presence of background traffic. The information available to the user is only the total number of jobs in each network, rather than the per-network numbers of foreground and background jobs. This leads to a complex partial information decision problem which is the focus of this chapter.

We develop and evaluate a Bayesian learning algorithm that splits a stream by optimally assigning entire jobs to different networks, such that the expected sojourn time is minimized. The algorithm learns as the system operates and provides information at each decision and departure epoch. We evaluate the optimality of the partial information algorithm by comparing the performance of the algorithm with the "ideal" performance obtained by solving a Markov decision problem with full state information. To this end, we have conducted extensive experiments both numerically and in a simulation tool that contains an implementation of the full wireless protocol stack. The results show that the Bayesian algorithm has close to optimal performance over a wide range of parameter values.

This chapter is based on the results presented in [19] and [20].

5.1 Introduction

In general, the more detailed information about the state of the system is known (e.g., the number of flows, measured round-trip times and the network load), the higher potential capacity improvements. However, in practice there is often no such detailed information available, or at best only some coarse-grained and aggregated statistics. Therefore, the challenge is to achieve efficient network utilization levels and good end-user application performance, based on informa-

tion that is only partially available. To address this challenge, we propose a dynamic Bayesian control algorithm that incorporates learning while optimally assigning the traffic to the different networks.

In this chapter we study the problem of optimizing the performance of file downloads over multiple networks in a queueing-theoretical context. Similar to Chapters 3 and 4, where N parallel networks, that are modeled as PS-queues, serve $N + 1$ streams of jobs. Stream 0 is called the foreground stream, and streams $1, \ldots, N$ are called the background streams. Jobs of background stream i are served exclusively at PS-queue i. However, the distinguishing feature of the concurrent access job-assignment model considered in this chapter is that each job of the foreground stream has to be assigned entirely to one of the PS-queues on the basis of information on the total number of (background and foreground) jobs at each of the queues. Job sizes are assumed to be exponentially distributed.

The goal is to develop a dynamic policy that minimizes the expected sojourn time of foreground jobs. Motivated by practice, we assume that the decision maker is not able to distinguish the number of foreground and background jobs in the network, but instead only has information on the total number of jobs. The challenge is to deal with this partial information problem. In this chapter, we address this problem through a learning mechanism, where the decision maker makes a statistical inference on the distribution of the numbers of foreground and background jobs after each decision. To this end, we model the system as a Bayesian decision problem. In this context, the decision making involves learning on the partially observed states while at the same time optimal actions are chosen. Experiments, conducted both numerically and in a simulation tool, show that the algorithm has a performance that is close to the case in which full information is available.

The main motivation for this chapter stems from applications of traffic optimizations in the context of multiple overlapping wireless networks. This application domain leads to a number of model assumptions that are not covered in the literature. First, an important aspect is the *presence of background traffic* (i.e., streams of jobs that are handled by one specific network only), which may have a strong impact on the performance and optimality of the assignment of the (foreground) stream of jobs; the inclusion of background traffic adds an interesting complexity to the model. Second, in practice background and foreground traffic flows can often not be separated, because the traffic from multi-homed systems is difficult to distinguish from the traffic in networks of single-homed systems, so that the only information available is the total number of jobs in the networks.

An optimal algorithm for assigning foreground jobs to the best PS-queue ideally has knowledge on the numbers of foreground and background jobs in the system. As a consequence, we have to deal with the availability of *partial information* (at best we only know the sum of the number of foreground and background jobs for each network), which adds another level of complexity to the model,

and requires smart methods to do so.

The contribution in this chapter is two-fold. On the methodological side, it is known that Bayesian learning algorithms are notoriously difficult to implement and to derive optimal policies from. While the vast majority of papers (see, e.g., [28, 30, 44, 97]) propose simplified structures of the optimal policy, we reduce the dimensionality of the state space by using structural properties of the problem. In addition, we show that efficient discretization of the state space leads to numerical tractability (see also Remark 5.2.1). On the practical side, the proposed algorithm is new in the context of concurrent access traffic optimizations for wireless networks. Such algorithms open up highly promising means to boost application performance over wireless networks. The algorithm is effective, yet easy to apply in practical systems, and as such extremely useful for real wireless network deployments.

In a queueing-theoretical context, there is very little literature on partial information models. Bellman [16] was the first to study decision problems with a transition law that is not completely known. He observed that the problem could be transformed into an equivalent full observation problem by augmenting the state space with the set of probability distributions defined on the domain of the unknown quantity (i.e., the unobserved state, or the unknown parameter) and updating it by Bayes' rule.

The transformation of the partial information problem to the complete information model, however, comes with added computational difficulties, since policies are defined over a continuum of states. This is the fundamental problem in developing algorithms for computing optimal policies [96]. There is some work in the theoretical domain to characterize the structure of the optimal policy (see, e.g., [29, 9, 117, 79]). Even then, finding the optimal policy computationally for a general Bayesian decision problem is intractable. Approaches dealing with this are to be satisfied with suboptimal solutions or to develop algorithms that can exploit problem characteristics (see, e.g., [77, 97, 125, 44, 28, 30]). We refer to [80, 89, 119, 73] for some surveys on computational techniques.

The organization of the chapter is as follows. In Section 5.2 we describe the model and introduce the notation. Moreover, we discuss the full information model in Section 5.2.1 and the Bayesian analysis in Section 5.2.2. In Section 5.3 we discuss the numerical results, comparing the performance of our Bayesian approach to the fully observable MDP, not only in a queueing-theoretical setting where networks are modeled as PS-queues, but also in a simulation setting where the full wireless protocol stack is implemented.

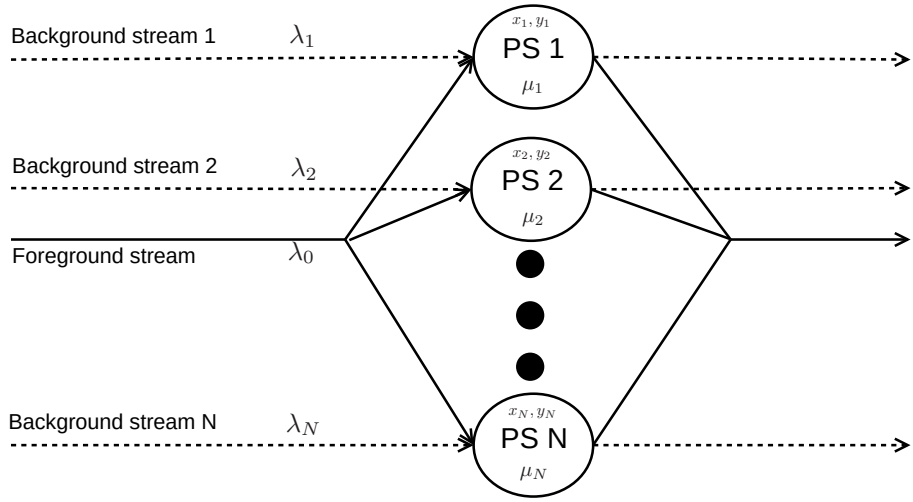


Figure 5.1: The concurrent access job-assignment model, where each of the N PS-queues represents a communication network that handles foreground and background file download traffic. In the concurrent access model, file downloads are modeled as jobs. Specifically for the job-assignment model, each foreground job is assigned entirely to one of the N available PS-queues. Background jobs of a certain stream are always processed by the same PS-queue.

5.2 Model description and analysis

In this section we describe the concurrent access job-assignment problem in greater detail. We model N mobile networks as PS-queues so that multiple jobs are served simultaneously. Accordingly, in our model we consider queue selection policies instead of network selection policies. There are $N + 1$ streams of jobs in the system. Stream i generates a stream of jobs for queue i for $i = 1, \ldots, N$. Stream 0 generates a stream of jobs for which the jobs can be sent to either queue 1 up to queue N. Hence, streams 1 to N can be seen as *background* traffic, and stream 0 as *foreground* traffic. We assume that all streams are modeled by independent Poisson processes with parameters $\lambda_0, \ldots, \lambda_N$. After a job enters the system, it demands service from the system. We assume that the service times follow an exponential distribution with mean service time $1/\mu_i = \beta_i$ for $i = 0, \ldots, N$. Then, the occupation rates ρ_i are defined by $\rho_i = \lambda_i \beta_i$. Note that for stability we require that the total load $\rho := \rho_1 + \cdots + \rho_N < N$. Based on the above information, one has to decide on the distribution of the foreground jobs over the N queues. The aim of the decision maker is to minimize the expected average number of foreground jobs in the system. Note that this is directly related to the sojourn times of the foreground traffic (see Figure 5.1). In the sequel we will study two dynamic models: the optimal queue selection model with full and partial observability. We first describe the full information model.

5.2.1 Full observation model

In this subsection we allow the decision maker to dynamically send the jobs to any queue. To find the optimal policy for making this decision, we model this as a Markov decision problem. To this end, let the state space $\mathcal{S} = \mathbb{N}_0^{2N} = \{0, 1, 2, \ldots\}^{2N}$. A tuple $s = (x_1, \ldots, x_N, y_1, \ldots, y_N) \in \mathcal{S}$ denotes that there are x_i foreground jobs and y_i background jobs at queue i for $i = 1, \ldots, N$. For each job, the set of actions is given by $\mathcal{A} = \{1, \ldots, N\}$, where $a \in \mathcal{A}$ denotes sending the job to queue a. When action a is chosen in state s, there are two possible events in the system; first, an arrival of a job can occur with rate λ_i or a job can finish his service with rate μ_i for $i = 0, \ldots, N$. The transition rates are thus given by $p = p(s, a, s')$, when the system is in state s, action a is taken and the next state is s', as follows:

$$p(s,a,s') = \begin{cases} \lambda_0, & \text{if } s' = s + e_a, \\ \lambda_i, & \text{if } s' = s + e_{i+N} \text{ for } i = 1, \ldots, N, \\ \mu_0, & \text{if } s' = s - e_i \text{ and } x_i > 0 \text{ for } i = 1, \ldots, N, \\ \mu_i, & \text{if } s' = s - e_{i+N} \text{ and } y_i > 0 \text{ for } i = 1, \ldots, N, \\ 0, & \text{otherwise,} \end{cases}$$

for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$, where e_i is the zero-vector with a one at the i-th entry. The term λ_0 represents an arrival of a foreground job that is assigned to queue a, λ_i represents an arrival of a background job at queue i, μ_0 represents the departure rates at all queue i in which foreground jobs are available, and μ_i represents the departure rate of a background job at queue i, if available. Since we are interested in the number of foreground jobs in the system, we take the cost function c equal to $c(s) = x_1 + \cdots + x_N$. The tuple $(\mathcal{S}, \mathcal{A}, p, c)$ defines the Markov decision problem.

Next, we uniformize the system (see Section 11.5 of [103]). To this end, we assume that the uniformization constant $\lambda_0 + \cdots + \lambda_N + \sum_{i=1}^{N} \max\{\mu_0, \mu_i\} = 1$; we can always get this by scaling. Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the rates to be transition probabilities. Define a *deterministic policy* π as a function from $\mathcal{S}$ to $\mathcal{A}$, i.e., $\pi(s) \in \mathcal{A}$ for all $s \in \mathcal{S}$. Note that the optimal policy can be found within this class (see [46]). Let $u_t^\pi(s)$ denote the total expected cost up to time t when the system starts in state s under policy π. Note that for any stable and work-conserving policy, the Markov chain satisfies the unichain condition, so that the average expected cost $g(\pi) = \lim_{t\to\infty} u_t^\pi(s)/t$ is independent of the initial state s (see Proposition 8.2.1 of [103]). The goal is to find a policy π^* that minimizes the long-term average cost, thus $g = \min_\pi g(\pi)$.

Let $V(s)$ be a real-valued function defined on the state space. This function will play the role of the relative value function, i.e., the asymptotic difference in total cost that results from starting the process in state s instead of some

reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) $g + V = TV$, where T is the dynamic programming operator acting on V defined as follows

$$\begin{aligned} TV(s) = &\sum_{i=1}^{N} x_i + \lambda_0 \min_{a\in\{1,\ldots,N\}} \{V(s+e_a)\} + \sum_{i=1}^{N} \lambda_i V(s+e_{i+N}) \\ &+ \sum_{i=1}^{N} \tfrac{x_i}{x_i+y_i}\mu_0 V(s-e_i) + \sum_{i=1}^{N} \tfrac{y_i}{x_i+y_i}\mu_i V(s-e_{i+N}) \\ &+ \Big(1 - \lambda_0 - \sum_{i=1}^{N} \big[\lambda_i + \tfrac{x_i}{x_i+y_i}\mu_0 + \tfrac{y_i}{x_i+y_i}\mu_i\big]\Big) V(s). \end{aligned} \tag{5.1}$$

The first term in the expression $TV(s)$ models the direct cost, the second term deals with the arrivals of foreground jobs, whereas the third term deals with the background jobs. The fourth and fifth terms concern service completions for foreground and background jobs, respectively. The last line is the uniformization constant.

The optimality equation $g + V = TV$ is hard to solve analytically in practice. Alternatively, the optimal actions can also be obtained by recursively defining $V_{l+1} = TV_l$ for arbitrary V_0. For $l \to \infty$, the maximizing actions converge to the optimal ones (for existence and convergence of solutions and optimal policies we refer to [103]). Note that for numeric computation the state space needs to be truncated to obtain a finite state space. In practice, one determines the truncation by systematically increasing the truncation bound until no significant changes in the average cost occur.

5.2.2 Bayesian partial information model

The dynamic queue selection model with full information uses a state description $(x_1, \ldots, x_N, y_1, \ldots, y_N)$ with $2N$ entries. However, in practice, distinguishing the foreground traffic from the background traffic might not be feasible. In these cases, one can only observe the state $(z_1, \ldots, z_N)$ with $z_i = x_i + y_i$ for $i = 1, \ldots, N$. Now, the dynamic control policy that we derived in the previous section cannot be applied straightforwardly. To apply the control policy one needs to create a mapping from $(z_1, \ldots, z_N)$ to $(x_1, \ldots, x_N, y_1, \ldots, y_N)$, so that (an estimate of the) full information is recovered. Note that it is not sufficient to create a mapping solely based on $(z_1, \ldots, z_N)$ at each decision epoch, since it does not use the information contained in the sample path, i.e., many sample paths can lead to the same state $(z_1, \ldots, z_N)$. Therefore, we will use Bayesian learning that takes into account the complete history of states in the estimation procedure. We shall call $z = (z_1, \ldots, z_N) \in \mathbb{N}_0^N$ the observation state. In order to learn about the division between the number of foreground and background jobs, we will denote by $u_i(n)$ the probability that at queue i there are n foreground jobs for $i = 1, \ldots, N$. The probability distribution u_i will serve

the purpose of information about the states that cannot be observed; hence, $u = (u_1, \ldots, u_N)$ is called the belief state. Note that the belief state space is of high dimension, namely $\prod_{i=1}^{N} \{u_i \in [0,1]^{\mathbb{N}_0} \mid \sum_{x \in \mathbb{N}_0} u_i(x) = 1\}$.

Based on the observation and belief states, we construct a state space for the Bayesian dynamic program consisting of the vectors $s = (z, u)$. Note that every arrival and departure gives the system information on how to update the belief state. Suppose that state s is given and that an arrival of a foreground job that is admitted to queue i occurs. The new state s_{af_i} is then given by $s_{af_i} = (z + e_i, u')$ where $u_i'(x) = u_i(x-1)$ for $x > 0$ and $u_i'(0) = 0$, and where $u_j'(x) = u_j(x)$ for $j \neq i$. In case of arrival of a background job to queue i, we have a new state $s_{ab_i} = (z + e_i, u)$.

In case of departures, we have a similar state transformation. When a foreground job leaves queue i, then we have corresponding states $s_{df_i} = (z - e_i, u')$. When a job leaves the system the type of departure is unknown, therefore the update equation will be a combination two prior belief states: $u_i'(x) = u_i(x)\frac{z-x}{z} + u_i(x+1)\frac{x+1}{z}$. If we believe there were $z - x$ background jobs before departure, then the probability a background job left equals $\frac{z-x}{z}$. The corresponding originating belief state probability is $u_i(x)$. Similarly, if we believe there were $x + 1$ foreground jobs before departure, then the probability a foreground left is $\frac{x+1}{z}$. The originating belief state probability is in this case $u_i(x+1)$. As not all transitions cannot be observed, we take the expectation with respect to the probability distribution u to average over all sample paths. This gives a new dynamic programming operator in which learning is incorporated. This is given by

$$\begin{aligned} TV(s) = \sum_{x_1 \in \mathbb{N}_0} \cdots \sum_{x_N \in \mathbb{N}_0} u_1(x_1) \cdots u_N(x_N) \Big[\sum_{i=1}^{N} x_i + \sum_{i=1}^{N} \lambda_i V(s_{ab_i}) + \\ \lambda_0 \min\{V(s_{af_1}), \ldots, V(s_{af_N})\} + \\ \sum_{i=1}^{N} \tfrac{x_i}{z_i} \mu_0 V(s_{df_i}) + \sum_{i=1}^{N} \tfrac{z_i - x_i}{z_i} \mu_i V(s_{db_i}) + \\ \Big(1 - \lambda_0 - \sum_{i=1}^{N} \big[\lambda_i + \tfrac{x_i}{z_i}\mu_0 + \tfrac{z_i - x_i}{z_i}\mu_i\big]\Big) V(s) \Big]. \end{aligned} \tag{5.2}$$

Note that the basic idea to transform (5.1) into (5.2) is to take the conditional expectation with respect to the belief state distribution u. Under this condition, the foreground and background jobs can be distinguished so that the structure of the equation resembles the one of the fully observed problem. However, only the transitions to the new belief state need to be adjusted so that the information that has been learned is taken into account. These transitions are provided above.

We end this section with two remarks.

Remark 5.2.1 (Complexity)**.** Note that the dynamic programming operator for the Bayesian model (5.2) resembles the dynamic programming operator of the full observation model (5.1). However, the state space of the Bayesian model is of significantly higher dimension as the state variables for the background traffic are continuous. Hence, solving the optimality equation $g + V = TV$ is notoriously hard, both analytically and numerically. In general, the Bayesian updates result in posterior distributions that cannot be captured by a nice structural form. In our problem, however, the decision maker can distinguish foreground and background jobs upon arrival leading to an arrival process with deterministic state transitions. It is only the departures that carry uncertainty with them. This leads to a state transition function, as described above, which keeps the dimensionality of the state space at reasonably low levels. In this way, the structure of the problem makes the Bayesian model a tractable approach (after discretization of the state space). Also note that for arbitrary queues i and j, the decision as to whether an incoming foreground job should join queue i or j, does not depend on the other queues. Hence, in the decision making one can compare queues 1 and 2, take the best queue and compare it to queue 3, take the best of that comparison and compare it to queue 4, and so forth. This leads to a sequence of $N - 1$ comparisons. Therefore, the Bayesian approach scales linearly in running time with the number of queues N.

Remark 5.2.2 (Accuracy)**.** In a general Bayesian setting, the belief state represents a probability distribution that represents the likelihood that the process is in a particular state. The accuracy of this estimate, generally, tends to deteriorate as the process progresses due to accumulated errors. In our problem setting, the accuracy of the estimates tends to improve as jobs leave the system. As more jobs leave the system, the support of the posterior distribution reduces to a smaller set of states, limiting the possibilities for errors. In fact, upon departure of the last job in a particular queue, the posterior distribution of that queue is independent of the past, since the state is exactly known. Thus, all probability mass is concentrated on having 0 jobs in that queue. Hence, an empty queue leads to a belief state that corresponds to the true state for that queue. This observation increases the accuracy of our algorithm due to stability of the system.

5.3 Numerical experiments

To assess the performance of our Bayesian algorithm for efficiently assigning downloads with concurrent access, we have performed extensive numerical experimentation, comparing the results of the Bayesian algorithm to the results of the fully observable MDP in which the foreground and background jobs can be distinguished. The comparison is done in two settings. Section 5.3.1 compares the performance of the full observation MDP with the Bayesian MDP under the model described in Section 5.2. Section 5.3.2 evaluates the performance of the

algorithms in a state-of-the-art wireless network simulation package in which the full wireless protocol stack is implemented. We have performed a large number of experiments with a wide range of parameter settings. The results are outlined below.

The number of model parameters is significantly large for already moderate numbers of queues, prohibiting extensive numerical experiments over the full spectrum of parameter combinations. To highlight the main effects of parameter changes, we considered two-queue scenarios, motivated by the fact that in the context of wireless networks, at most a few parallel networks will be used simultaneously per user. Moreover, the file-size distributions were taken to be exponential (see also the remarks in Section 5.4) with means $\beta_0 = \beta_1 = \beta_2 = 1$. To allow for asymmetric network settings, we scale the job-size distribution in queue i by C_i, i.e., the effective job size is therefore β_i/C_i for queue $i = 1, 2$. To include scenarios for light and heavy foreground traffic, the foreground traffic $\rho_0 = \lambda_0\beta_0/\min\{C_1, C_2\}$ was varied as 0.1 and 0.9, and the background loads $\rho_1 = \lambda_1\beta_1/C_1$ and $\rho_2 = \lambda_2\beta_2/C_2$ were varied between 0.1 and 0.9.

5.3.1 Comparison of the fully observed MDP against the Bayesian algorithm

To compare the quality of the Bayesian approach discussed above to the fully observed MDP approach, we have calculated the mean of the sojourn time S of an arbitrary job for both policies, under a variety of parameter settings. To this end, for each scenario we have calculated the mean number of jobs, following the lines of Sections 5.2.1 and 5.2.2 and then used Little's formula to obtain $\mathbb{E}S$. Denoting by $\mathbb{E}[S|\text{Bayes}]$ and $\mathbb{E}[S|\text{full MDP}]$ the expected sojourn times under the Bayesian approach and the full MDP approach, respectively, the relative difference is defined as follows

$$\Delta\% = \frac{\mathbb{E}[S|\text{Bayes}] - \mathbb{E}[S|\text{full MDP}]}{\mathbb{E}[S|\text{full MDP}]} \times 100\%. \tag{5.3}$$

Note that the simulations have been run with 10^7 foreground jobs resulting in a 99% confidence interval of approximately 0.1% with respect to the point estimates.

Equal network capacities
Table 5.1 shows the results for light foreground load (with $\rho_0 = 0.1$) and for a variety of background-load values (ρ_1, ρ_2) for the case of symmetric networks, and where the network capacities are equal (normalized to $C_1 = C_2 = 1$). More specifically, for each scenario, Table 5.1 shows the triple
$(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$, where $\Delta\%$ is defined in (5.3). Note that, due to symmetry, only the results for $\rho_2 \geq \rho_1$ are shown. Table 5.2 and Figure 5.2 show the results for medium to heavy foreground load (with $\rho_0 = 0.9$).

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.9
0.1	(1.073, 1.072, 0.2%)	(1.116, 1.115, 0.1%)	(1.165, 1.164, 0.1%)	(1.210, 1.210, 0.0%)	(1.245, 1.241, 0.4%)
0.2		(1.196, 1.195, 0.1%)	(1.273, 1.271, 0.2%)	(1.352, 1.350, 0.1%)	(1.416, 1.409, 0.5%)
0.3		(1.282, 1.277, 0.4%)	(1.393, 1.390, 0.2%)	(1.516, 1.514, 0.1%)	(1.628, 1.625, 0.2%)
0.4			(1.522, 1.519, 0.2%)	(1.715, 1.711, 0.2%)	(1.918, 1.911, 0.3%)
0.5			(1.674, 1.665, 0.5%)	(1.958, 1.952, 0.3%)	(2.318, 2.308, 0.4%)
0.6				(2.260, 2.255, 0.3%)	(2.910, 2.897, 0.5%)
0.7				(2.654, 2.641, 0.5%)	(3.878, 3.858, 0.5%)
0.8					(5.735, 5.705, 0.5%)
0.9					(11.064, 11.020, 0.4%)

Table 5.1: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ for foreground load $\rho_0 = 0.1$.

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.9
0.1	(1.59, 1.55, 2.3%)	(1.93, 1.88, 2.6%)	(2.53, 2.46, 2.9%)	(3.76, 3.70, 1.7%)	(8.90, 8.89, 0.2%)
0.2		(2.21, 2.15, 2.9%)	(3.08, 2.99, 2.8%)	(5.31, 5.23, 1.5%)	unstable
0.3		(2.60, 2.51, 3.5%)	(3.99, 3.86, 3.4%)	(9.74, 9.63, 1.1%)	unstable
0.4			(5.67, 5.60, 1.3%)	unstable	unstable
0.5			(10.87, 10.62, 2.4%)	unstable	unstable

Table 5.2: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ for foreground load $\rho_0 = 0.9$.

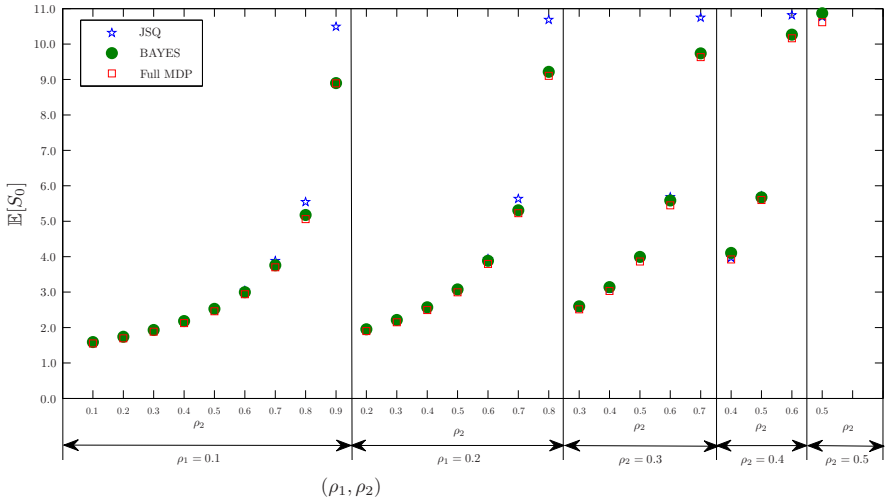


Figure 5.2: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \mathbb{E}[S|\text{JSQ}])$ for foreground load $\rho_0 = 0.9$.

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.9
0.1	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.514, 0.513, 0.2%)	(0.742, 0.741, 0.2%)	(1.067, 1.062, 0.5%)
0.2	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.516, 0.516, 0.1%)	(0.767, 0.766, 0.1%)	(1.167, 1.162, 0.4%)
0.3	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.519, 0.519, 0.0%)	(0.792, 0.792, 0.1%)	(1.286, 1.281, 0.4%)
0.4	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.521, 0.521, 0.1%)	(0.816, 0.815, 0.1%)	(1.424, 1.419, 0.3%)
0.5	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.523, 0.523, 0.0%)	(0.838, 0.837, 0.1%)	(1.593, 1.587, 0.4%)
0.6	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.525, 0.525, 0.0%)	(0.858, 0.858, 0.1%)	(1.800, 1.794, 0.3%)
0.7	(0.286, 0.286, 0.0%)	(0.370, 0.370, 0.0%)	(0.525, 0.525, 0.0%)	(0.875, 0.874, 0.1%)	(2.050, 2.044, 0.3%)
0.8	(0.286, 0.286, 0.0%)	(0.371, 0.371, 0.0%)	(0.526, 0.526, 0.0%)	(0.891, 0.891, 0.0%)	(2.366, 2.360, 0.2%)
0.9	(0.285, 0.285, 0.0%)	(0.371, 0.371, 0.0%)	(0.526, 0.526, 0.0%)	(0.901, 0.901, 0.0%)	(2.773, 2.769, 0.1%)

Table 5.3: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ for foreground load $\rho_0 = 0.1$ with $C_1 = 1$ and $C_2 = 4$.

Tables 5.1 and 5.2 reveal that the Bayesian algorithm has very good performance that is close to a system with full information for a wide range of parameter combinations of background loads (ρ_1, ρ_2). This is remarkable since the Bayesian algorithm has less information available than the full observation MDP, but learns sufficiently to make good decisions. Note that Table 5.1, with $\rho_0 = 0.1$, has a performance that is extremely close to the full observation model, with errors typically less than 0.5%. Table 5.2, with $\rho_0 = 0.9$, has a slightly degraded performance, with errors typically less than 3.5%, over a broad range of parameters settings. Nonetheless, the Bayesian algorithm clearly outperforms the widely used Join the Shortest Queue (JSQ) algorithm, see Figure 5.2. The slightly degraded performance of the Bayesian algorithm is due to the fact that the Bayesian algorithm has estimates of the probability distribution on the foreground and background traffic with high accuracy as more and more jobs leave the system. When the load moves from 0.1 to 0.9 the time it takes to return to more accurate estimates is longer, which explains the slightly degraded performance (see also Remark 5.2.2).

Unequal network capacities

To assess the usefulness of the Bayesian approach to the case of unequal network capacities, we have also performed experiments for the case $C_1 \neq C_2$. Table 5.3 below shows the results of the model considered in Table 5.1, but with the (normalized) network capacities for $C_1 = 1$ and $C_2 = 4$. Similarly, Table 5.4 shows the results for the models considered in Table 5.2, with network capacities $C_1 = 1$ and $C_2 = 4$.

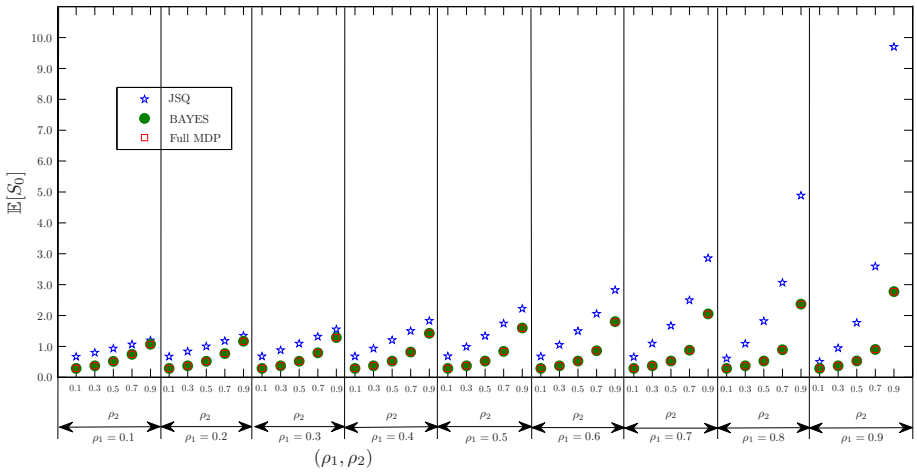


Figure 5.3: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \mathbb{E}[S|\text{JSQ}])$ for foreground load $\rho_0 = 0.1$ with $C_1 = 1$ and $C_2 = 4$.

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.9
0.1	(0.369, 0.366, 0.8%)	(0.498, 0.493, 1.0%)	(0.704, 0.701, 0.39%)	(1.127, 1.115, 1.00%)	(2.867, 2.745, 4.4%)
0.2	(0.369, 0.367, 0.6%)	(0.501, 0.498, 0.6%)	(0.734, 0.722, 1.61%)	(1.208, 1.197, 0.90%)	
0.3	(0.369, 0.368, 0.3%)	(0.511, 0.504, 1.3%)	(0.754, 0.745, 1.20%)	(1.300, 1.290, 0.80%)	
0.4	(0.370, 0.369, 0.3%)	(0.514, 0.509, 0.9%)	(0.775, 0.769, 0.81%)		
0.5	(0.370, 0.369, 0.2%)	(0.516, 0.514, 0.5%)	(0.797, 0.792, 0.60%)		
0.6	(0.370, 0.370, 0.1%)	(0.521, 0.518, 0.6%)			
0.7	(0.370, 0.370, 0.1%)	(0.522, 0.521, 0.3%)			
0.8	(0.370, 0.370, 0.0%)				
0.9	(0.370, 0.370, 0.0%)				

Table 5.4: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ for foreground load $\rho_0 = 0.9$ with $C_1 = 1$ and $C_2 = 4$.

Tables 5.3 and 5.4 show again that the Bayesian algorithm has excellent performance, doing much better than JSQ (see Figure 5.3). Under light foreground traffic load ($\rho_0 = 0.1$), the error is extremely small (0.5% or far less). When the foreground traffic load is increased to $\rho_0 = 0.9$ the error remains small with typical values of 1% up to a maximum of 4.4% for an unbalanced system.

Comparing the results with the results of the symmetric network capacity settings in Tables 5.1 and 5.2, we observe that the relative performance in the asymmetric case of the Bayesian algorithm is even better, especially when the foreground load is significant. This may be explained by the fact that in the asymmetric system with $C_2 = 4$, the Bayesian algorithm has more observations to learn from due to the increased number of departures in the queue 2, and because an erroneous decision has little impact due to short busy periods in queue 2. Only when queue 2 is heavily loaded, the system performance is extremely sensitive to the proper scheduling actions, which explains the maximum error in Table 5.4.

In conclusion, the results in this section show that the Bayesian algorithm has a performance extremely close to the performance of a fully observed MDP both under symmetric and asymmetric systems.

5.3.2 Comparison of the fully observed MDP against the Bayesian algorithm in wireless networks

To evaluate the accuracy of the Bayesian approach in a realistic wireless networking environment, we have performed extensive experimentation in OPNET Modeler [91] that implements the full protocol stack of wireless equipment. We emphasize that this is not a trivial experiment, because the Bayesian learning and decision functionality must be embedded in the network terminals and the complex mapping of the physical and medium access control layer parameters to the PS-queue parameters must be performed. In this context, it was shown in Chapter 2 based on extensive network simulations how the download performance of TCP-based wireless networks can be modeled by PS-queues, by using the proper parameter mapping. In our experiments, the simulated response-time performance of TCP-based networks under the Bayesian approach was benchmarked against the simulated response-time performance under the fully observable MDP traffic assignment. The results are outlined below.

Experimental configuration

Figure 5.4 shows the network topology in which the traffic assignment solution operates. In practice, all wireless terminals may download files from an application server, which may also be a dispatcher in front of several application servers (not shown). The application server is considered to have information about the number of ongoing downloads over each of the WLAN access networks, AP1 and AP2, but is unable to distinguish between the multi-homed and the single-homed

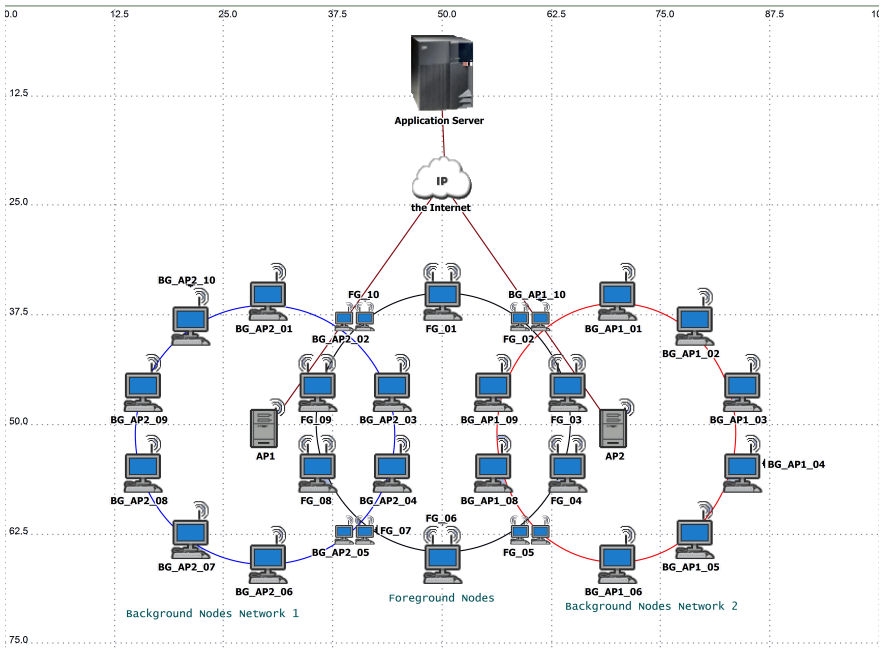


Figure 5.4: The concurrent access network topology.

terminals, because there is no binding between both network addresses of the multi-homed terminals. Both WLAN access points operate on non-overlapping frequency channels to establish two non-interfering parallel paths to the application server from the multi-homed systems. The transmission links from the access points towards the application server are considered to incur no delay nor any loss to packets from and to the access points. This assumption is motivated by the much higher capacities and reliability offered in contemporary fixed-line carrier-grade Internet connections in comparison to the IEEE 802.11b access networks. For sake of simplicity, the decision logic was placed in each foreground terminal and has instant knowledge on the number of flows in each network.

The analytic model from Chapter 2 captures the combined dynamics and protocol overhead of the 802.11 MAC, IP, TCP and application-layer into an explicit expression for the effective service time of a file download. Based on the effective service time, the effective load can be determined of the file transfers in our simulated WLAN networks with a flow-level $M/G/1$-PS model.

In the simulated network there are ten multi-homed terminals (named $FG\_01 - FG\_10$) that generate download requests (that are considered foreground jobs in the queueing model) with arrival rate λ_0. These foreground terminals are

positioned between both access points in a circle with a radius of 15 meter. In addition there are ten single-homed terminals (named with prefix $BG\_AP1\_$) that generate background traffic in network 1 with file downloads arriving with rate λ_1 to the first network. The remaining ten single-homed terminals (named with prefix $BG\_AP2\_$) generate background traffic at rate λ_2 in network 2 in a similar fashion. All background terminals are positioned at an equal distance of 15 meter from their respective access point. The file download requests arrive according to an independent Poisson process and terminals may have multiple file transfers in progress.

The MAC/PHY parameters of the WLAN stations are set in accordance with the widely deployed IEEE 802.11b standard amendment as it relies on the same MAC protocol basis as the contemporary higher rate (IEEE 802.11 a/g/n) amendments and has lower computational requirements for high-load network simulations. Table 5.5 summarizes the IEEE 802.11 MAC parameters used in our analytic model to calculate the effective load values for the simulation runs.

parameter	value	parameter	value
mac	224 bits	ack	112 bits
difs	50 μs	R_b	$\{1, 11\} \cdot 10^6$ bps
sifs	10 μs	$Cw_{\min}$	31 slots
eifs	364 μs	phy	192 μs
δ	1 μs	τ	20 μs
R_c	10^6 bps		

Table 5.5: IEEE 802.11b MAC parameters.

In this table, mac is the number of bits of overhead bits associated with a MAC data frame. The difs, sifs, eifs are the DCF, short and extended interframe spacing times, respectively. The δ is the propagation delay that is assumed in our analytic model. R_c is the transmission rate for WLAN acknowledgments of size ack bits, and R_b is the WLAN transmission rate for MAC data frames that is set to 1 or 11 Mbps. $Cw_{\min}$ corresponds to the minimum contention window in slots. Phy is the physical layer overhead, and τ is the slot time. In addition to the WLAN MAC, specific settings apply to the higher protocol layers and are outlined in Table 5.6.

variable	setting
X_{FTPget}	4096 bits
$X_{FTPclose}$	64 bits
TCP_{stack}	Full-Featured
X_{MSS}	11584 bits
$X_{tcp/ip}$	416 bits
w	70080 bits (8760 bytes)
X_{file}	$1.6 \cdot 10^6$ bits

Table 5.6: Network and application settings.

In Table 5.6, X_{FTPget} is the size of the FTP GET-command that is issued for initiating a file download, $X_{FTPclose}$ is the size of the FTP CLOSE-command that concludes the file transfer at the application. The TCP stack used in our experiments is characterized in OPNET as 'Full-Featured' that has an MSS equal to X_{MSS} (in bits). The number of TCP/IP overhead bits per segment is $X_{tcp/ip}$ bits. The maximum TCP receiver window size is indicated as w (in bits), and the file size as X_{file} (in bits).

Note that the parameter settings from Tables 5.5 and 5.6 correspond to one of the parameter sets (for one of the WLAN transmission rates for MAC data frames, R_b) of the model validation experiments in Chapter 2, of which the results are shown in Figure 2.4b. When respecting the engineering guidelines from Section 2.5 we can assume that the mean download response times in our simulation model can be accurately predicted from the effective load of the network using the $M/G/1$-PS model.

Experimental results

To assess the performance of the Bayesian algorithm in practice, we have performed simulations with the OPNET Modeler [91] with the full wireless protocol stack implemented for both the fully observable MDP and the Bayesian model. The OPNET simulations for the experimental results have been run with approximately $322,000$ foreground jobs and the background jobs ranging from roughly $644,000$ jobs to 5.1 million jobs depending on the load. In our simulation study we have considered two scenarios. One simulation scenario considers equal capacity networks in which all terminals are configured to use a WLAN transmission rate of 11 Mbps. For simulating a scenario in which the network capacity of both access network is unequal, the WLAN transmission rate used in AP2 is lowered to 1 Mbps, which reduces the medium capacity for processing file transfers by a factor of 5.79. In this scenario, the background load applied to AP2 is based on the lower capacity, whereas the foreground traffic intensity remains the same as for the equal capacity network.

We have executed 48 runs for the equal capacity scenario (24 runs for the fully

observed MDP and 24 runs for the Bayesian MDP) and 80 runs for the unequal capacity scenario. All runs have completed a total simulation time of 300 hours per run of which 1 hour is the warm-up time leading to a wall clock time of approximately 75 hours per run. This experimental setup is sufficient to derive a 99% confidence interval of approximately 0.7% with respect to the point estimates. The results of the experiments for both scenarios are outlined in Tables 5.7 and 5.8 for $\rho_0 = 0.1$ and a number of combinations ρ_1 and ρ_2. Note that the load values in this setting are obtained following the parameterization as defined and validated in Chapter 2. The results in Tables 5.7 and 5.8 show

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.8
0.1	(0.355, 0.354, 0.31%)	(0.370, 0.369, 0.30%)	(0.385, 0.385, 0.05%)	(0.402, 0.400, 0.49%)	(0.408, 0.406, 0.46%)
0.2		(0.396, 0.395, 0.21%)	(0.420, 0.420, 0.10%)	(0.446, 0.446, 0.05%)	(0.456, 0.455, 0.11%)
0.3		(0.421, 0.421, 0.18%)	(0.460, 0.458, 0.33%)	(0.502, 0.498, 0.65%)	(0.521, 0.519, 0.48%)
0.4			(0.503, 0.501, 0.35%)	(0.565, 0.564, 0.18%)	(0.601, 0.599, 0.37%)
0.5			(0.551, 0.547, 0.61%)	(0.643, 0.639, 0.68%)	(0.700, 0.696, 0.53%)
0.6				(0.742, 0.737, 0.68%)	(0.831, 0.828, 0.29%)
0.7				(0.867, 0.865, 0.20%)	(1.028, 1.018, 1.01%)
0.8					(1.322, 1.319, 0.23%)

Table 5.7: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ in OPNET for $\rho_0 = 0.1$.

ρ_1 \ ρ_2	0.1	0.3	0.5	0.7	0.8
0.1	(0.417, 0.415, 0.4%)	(0.415, 0.415, 0.0%)	(0.416, 0.416, 0.0%)	(0.416, 0.416, 0.0%)	(0.415, 0.415, 0.0%)
0.2	(0.476, 0.474, 0.4%)	(0.477, 0.475, 0.3%)	(0.479, 0.476, 0.6%)	(0.475, 0.475, 0.0%)	(0.475, 0.475, 0.0%)
0.3	(0.556, 0.555, 0.2%)	(0.556, 0.555, 0.3%)	(0.553, 0.551, 0.4%)	(0.557, 0.556, 0.2%)	(0.555, 0.555, 0.0%)
0.4	(0.663, 0.660, 0.4%)	(0.665, 0.664, 0.3%)	(0.667, 0.666, 0.2%)	(0.667, 0.666, 0.1%)	(0.667, 0.664, 0.4%)
0.5	(0.807, 0.806, 0.1%)	(0.819, 0.818, 0.1%)	(0.827, 0.826, 0.1%)	(0.830, 0.829, 0.1%)	(0.835, 0.833, 0.2%)
0.6	(1.016, 1.013, 0.3%)	(1.052, 1.047, 0.5%)	(1.068, 1.065, 0.3%)	(1.083, 1.081, 0.2%)	(1.100, 1.099, 0.0%)
0.7	(1.322, 1.305, 1.3%)	(1.410, 1.390, 1.4%)	(1.482, 1.476, 0.4%)	(1.546, 1.543, 0.2%)	(1.597, 1.595, 0.1%)
0.8	(1.817, 1.777, 2.3%)	(2.057, 2.013, 2.2%)	(2.299, 2.273, 1.2%)	(2.675, 2.610, 2.5%)	(2.876, 2.862, 0.5%)

Table 5.8: Comparison of $(\mathbb{E}[S|\text{Bayes}], \mathbb{E}[S|\text{full MDP}], \Delta\%)$ in OPNET for $\rho_0 = 0.1$ with $C_1 = 1$ and $C_2 = 0.17$.

that the Bayesian approach again closely matches the performance of the full observation MDP model in a real networking environment for equal and non-equal capacity networks. This is most remarkable, since the OPNET Modeler is a packet-level simulator implementing the full protocol stack including TCP, IP, MAC, and PHY layers, whereas both the fully observable and the Bayesian MDP are based on a processor-sharing model for the networking environment. The results also illustrate that the Bayesian approach is a very powerful means that is practically applicable.

5.4 Conclusions and further research

In this chapter we study a model in which the sojourn time of foreground traffic is minimized in the presence of background traffic with the restriction that the different traffic types cannot be distinguished. We propose to adopt a Bayesian

methodology to control the system. Our results shows that even though the system has fewer information than a fully observed system, the partial observability does not significantly compromise foreground sojourn times. Practically, the Bayesian setting is better applicable to a multi-network environment as full system observability cannot be assumed. The Bayesian method is very robust under different parameter settings.

The results raise a number of interesting questions for further research. First, in the current model we assume that the job sizes are exponentially distributed. An interesting question is how the Bayesian algorithm performs under more general job size distributions. For this purpose, our MDP approach could be extended by modeling the job size distributions by phase-type distributions, adding more challenges to the computational burden. Note that this comes with additional questions regarding the observability of the number of jobs in the different phases. Second, despite the fact that in practical deployments N is likely to be small, it is of theoretical interest to evaluate the performance and complexity of the Bayesian algorithm for larger N. Third, in this chapter we primarily focused on the optimization of the foreground traffic, whereas in reality the performance of the background traffic may also be subject to QoS requirements. Inclusion of such QoS constraints addresses an interesting and practically important topic.

Chapter 6

Dynamic Traffic Splitting over Multiple Networks

In the previous chapters the static splitting and dynamic assignment of traffic flows is considered for improving the mean sojourn time of these flows in the presence of background traffic in an environment where multiple PS-queues are available to process these flows. The Concurrent Access job-split and job-assignment models consist of multiple PS-queues that each represent a wireless network.

In this chapter dynamic traffic splitting of flows over multiple PS-queues is considered. To analyze the performance of dynamic traffic splitting, the concurrent access dynamic job-split model is developed for "optimal" traffic splitting, where dynamic splitting based on full state information is performed at infinitely fine-grained granularity. Next, we present a practical realization that applies a splitting algorithm for TCP networks that uses a simple score function to make on-the-fly decisions on the routing of individual TCP segments, based on the measured per-connection RTT, transmission-buffer content and throughput. Then, we use a combination of the effective service time model from Chapter 2 and our concurrent access model as a benchmark to evaluate the efficiency and practical usefulness of the practical realization of dynamic splitting TCP flows over real wireless networks in a test-lab environment.

Extensive experimentation demonstrates that our solution is extremely efficient and easily deployable, and as such provides a powerful means to effectively split TCP traffic in the presence of concurrently available access networks.

This chapter is based on the results presented in [54] and [55].

6.1 Introduction

In this chapter a concurrent access network is considered that is similar to the ones studied in the previous three chapters. In Chapters 3 and 4 streams of

traffic are split into fragments according to a static splitting rule that is based on the average load of the available processing queues. Subsequently, Chapter 5 has studied a dynamic traffic assignment approach that is based on the actual state of the available processing queues.

Both approaches have their pros and cons. On the one hand, splitting traffic may significantly reduce the processing time of a flow when compared to assigning flows. The individual sojourn times may be as much as i times smaller those resulting from assignment over i available queues. On the other hand, splitting may also adversely affect the performance because it creates a dependency between all fragments that are processed in parallel. This dependency adversely affects the performance of an entire flow when one of its fragments meets the unfavorable circumstance of being processed by a queue in which the actual traffic load is much higher than expected. Dynamic decision algorithms may mitigate such unfavorable circumstances by incorporating full or partial state information from the processing queues. By basing decisions on actual state information, short-term under and over utilizations of queues may be observed and accounted for in the decisions taken and benefited from.

It is important to distinguish dynamic assignment from dynamic splitting. The first type of algorithms may select the best queue at a specific point in time. If suddenly the circumstances in the chosen queue become unfavorable (e.g., due to background traffic), the performance of the ongoing flows may be lower than if another available queue was selected. When considering an individual flow, there is no dynamic adaptation to changing circumstances in the available queues during the processing of the flow. The latter type of algorithms are, in contrast, fully dynamic by changing the splitting ratio during the processing of the flow's fragments at the queues. As a result, ideal dynamic traffic splitting algorithms are expected to outperform their dynamic assignment counterparts.

Despite the applicability of PS-based models to real communication networks, there is little known on PS-based models suitable for modeling the use of multiple networks concurrently. As an exception, Key et al. investigate the efficiency of combining multipath routing and congestion control in TCP-based networks. In [68] they show that under certain conditions the allocation of flows to paths is optimal and independent of the flow control algorithm used. In [69] it is shown that with RTT bias uncoordinated control can lead to inefficient equilibria, while without RTT bias, both coordinated and uncoordinated Nash equilibria correspond to desirable welfare maximizing states. The distribution and reassembly of tasks are typically modeled by fork-join constructions [71], in many cases embedded in so-called stochastic activity networks. In cases where the processing times of the subtasks are independent, exact or numerical analysis is relatively simple (e.g., [32]), whereas the inclusion of dependent processing times (e.g., due to queueing or job splitting) typically leads to very complex analysis (e.g., [37, 81]) and no closed-form solution exists. For PS-based nodes that process the tasks of a job in parallel, the complex correlation structure

between the sojourn times at the PS nodes makes an exact detailed mathematical analysis of the model impossible. In [74] and [75], the author analyzes a similar model but with FCFS queues and with probabilistic splitting. We further refer to Altman et al. [11], who consider routing policies in a distributed versus centralized environment. In general our queueing model falls within the framework of fork-join queueing networks, see [12] for an extensive overview. In Chapter 3, the theoretical foundation for a tail-optimal splitting rule is provided for light foreground load that is shown to work well with respect to both the tail asymptotics and the mean sojourn times.

Despite the fact that the analytic models in the literature provide important and valuable insight in the performance implications and efficiency of traffic-splitting algorithms, they do not explicitly reveal how to exploit contemporary protocol implementations on CA in practical deployments. Motivated by this, the aim of this chapter is to propose and evaluate a simple, yet efficient and easily deployable traffic-splitting algorithm for TCP-based networks. To this end, we first propose a model that "optimally" splits traffic in a dynamic way based on full state information at infinitely fine granularity. This model is called the concurrent access network (CAN) model and will be used as a benchmark. We show that the expected response time under this "optimal" splitting performance can be numerically calculated by solving a continuous-time Markov chain. In practical deployments, however, there is only limited and coarse-grained information (e.g., measured RTTs, queue lengths and throughputs) is available to base routing decisions on, so that this optimal performance - which will be used as a benchmark - can not be achieved. Next, motivated by the work in [43], we propose a simple and easily deployable method for TCP-level traffic splitting, where TCP segments are routed based on a score-function that dynamically adapts the routing decisions to observed per-node RTTs, transmission-buffer contents and measured throughput values. Then, we present the results of extensive test-lab experiments to assess the effectiveness of our approach. The results show that the score-function method matches the performance of the benchmark model extremely well for a wide range of parameter settings, and as such provides a powerful means to effectively split TCP traffic in the presence of concurrently available access networks.

The remainder of this chapter is organized as follows. In Section 6.2 we describe the CAN model and introduce the notation. In Section 6.3 we assess the "optimal" performance of the CAN-model by simulations, and show that the expected response time under the CAN-model is nearly insensitive to the job-size distribution. This allows us to numerically calculate the optimal response-time performance by solving the steady-state distribution of a continuous-time Markov chain. In Section 6.4 we propose a simple score-function based approach to dynamically split traffic at the TCP-layer over different networks. In Section 6.5 we discuss the results of extensive experimentation in a test-lab environment. Finally, in Section 6.7 we address a number of topics for further research.

6.2 Concurrent access network model

The CAN model consists of N parallel PS nodes. There are $N+1$ traffic streams: a single stream of foreground jobs (called class-0 jobs) and N streams of background jobs (called class-i jobs, for $i = 1, \ldots, N$). Class-i jobs arrive according to independent Poisson processes with rates λ_i, the service times are generally distributed with mean β_i, and the corresponding load offered to the system is $\rho_i = \lambda_i \beta_i$, $i = 0, 1, \ldots, N$. Foreground jobs use the capacity of all N nodes simultaneously in a fluid-like manner, using the (instantaneously) available capacity at all the N PS nodes; at any moment in time the capacity available at node i is equally shared amongst all foreground jobs in the system and with the background jobs at node i. The splitting operates without delay with infinitely small granularity and has perfect information about the number of foreground and per-class background jobs in the system. If upon arrival of a tagged foreground job F there are k_0 other foreground jobs in the system and k_i background jobs at node i, then F obtains a fraction

$$f_i := \frac{1}{k_0 + 1 + k_i} \tag{6.1}$$

of the capacity of node i, for $i = 1, \ldots, N$. Note that in this way, the instantaneous total transmission speed that F receives equals $\sum_{i=1}^{N} f_i$, and that this speed changes during the course of the sojourn time of F in the system, as other jobs may come and go. Also, it should be noted that using this fluid-like splitting of foreground jobs is "optimal" in the sense that the synchronization delay (which is usually encountered when splitting is done at coarse-grained granularity or with non-perfect or delayed information) is zero, while the foreground jobs receive no more than their fair share of capacity at each of the nodes; it is evident that even better performance for foreground jobs can be obtained by allowing unfair capacity sharing at the PS nodes in favor of foreground traffic (see also Section 6.7).

If we assume that the service-times are exponentially distributed with rates $\mu_i := 1/\beta_i$ $(i = 0, 1, \ldots, N)$, then the evolution of the system can be described as a continuous-time Markov chain (CTMC) with state space $S = {\mathbb{N}_0}^{N+1}$, where each state is of the form $s = (k_0, k_1, \ldots, k_N) \in S$, with k_0 the number of foreground jobs in all N PS nodes and k_i $(i = 1, \ldots, N)$ the number of background jobs in PS node i. For each arriving foreground job, a task is assigned to each PS node of which the processing demand will be adjusted to complete the service of all tasks corresponding to the same job simultaneously. It is readily verified that the state-transition rates of the CTMC are as follows:

$$q(s, s + e_i) = \lambda_i \quad (i = 0, 1, \ldots, N), \tag{6.2}$$

$$q(s, s - e_0) = \sum_{i=1}^{N} \frac{k_0}{k_0 + k_i} \mu_0, \tag{6.3}$$

$$q(s, s - e_i) = \frac{k_i}{k_0 + k_i}\mu_i \quad (i = 1, \ldots, N), \tag{6.4}$$

for all possible state combinations in S; here, e_i stands for the unit vector that has zeros in all components except the component that corresponds to the total number of foreground jobs (by taking $i = 0$) or to the number of background jobs (for $i = 1, \ldots, N$). Here (6.2) represents the external arrivals of class-i jobs, for $i = 0, 1, \ldots, N$. The departure of a foreground job, and a class-i job are represented by (6.3) and (6.4) respectively. Given the stationary distribution of the CTMC, $\pi(\cdot)$, the expression for the expected number of foreground jobs in the system is:

$$\mathbb{E}[N_0] = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_N=0}^{\infty} i_0 \pi(i_0, i_1, \ldots, i_N), \tag{6.5}$$

where $\pi(.)$ denoted the stationary distribution of our Markov process. Using Little's formula, we obtain the expected sojourn time of the foreground jobs:

$$\mathbb{E}[S_0] = \frac{\mathbb{E}[N_0]}{\lambda_0}. \tag{6.6}$$

6.3 Analysis of the CAN-model

In Section 6.2 it was assumed that the job-size distributions are exponentially distributed. However, in practice, the job-size distribution in TCP-based networks is usually far from exponential. Motivated by this, in this section we analyze the impact of the job-size distributions on the performance of the system, using simulations. To this end, we have coded a simulation program that optimally splits the traffic among two PS nodes based on full state information, and at infinitely fine-grained granularity, leading to "optimal" performance, and in particular minimizes $\mathbb{E}S_0$, for any choice of the job-size distribution.

We have performed extensive simulations to obtain $\mathbb{E}S_0$ for a wide variety of scenarios. The job-size distributions were varied as deterministic (with squared coefficient of variation $c_B^2 = 0$), exponential ($c_B^2 = 1$), hyperexponential (with balanced means and squared coefficient of variation $c_B^2 = 16$), Pareto-3 (i.e. with $Pr\{B > x\} = (1 + x/2)^{-3}$, $x \geq 0$, and hence $c_B^2 = 3$) and Pareto-2 (i.e., with $Pr\{B > x\} = \frac{1}{4x^2}$, $x \geq \frac{1}{2}$, and hence $c_B^2 = \infty$), all with mean 1. To limit durations of the simulation runs, we considered scenarios with $N = 2$ networks, which is the most common CA-scenario in practical deployments. Also, to limit the number of scenarios, in all cases considered the squared coefficients of variation of the job-size distributions were taken to be the same for all classes (i.e., $c_{B_i}^2 = c_B^2$ for $i = 0, 1, \ldots, N$). The foreground load ρ_0 was varied as 0.1, 0.9 and 1.8 to cover load values ranging from light to heavy foreground load. The background-load combinations (ρ_1, ρ_2) were varied such that ρ_1 and ρ_2 take values 0.1, 0.3, 0.5, 0.7 and 0.9 (of course, only for the parameter combinations for which the system is stable), hence covering scenarios with both light- and

heavy-traffic and varying degrees of asymmetry. The results of our experiments are outlined below. All simulation runs are based on at least 10^8 (10^9 for very high load values) foreground observations, leading to accurate predictions of $\mathbb{E}S_0$, such that all digits shown below are significant; confidence intervals have been omitted for compactness of the presentation. In addition to the simulations, we have also calculated the values of $\mathbb{E}S_0$ numerically for the special case of exponential job-size distributions by using (6.1)-(6.6); truncation of the state space was done at a sufficiently large size in order not to influence the results. Table 6.1 shows the results for lightly loaded foreground traffic, with $\rho_0 = 0.1$. To avoid duplication due to symmetry, the results are shown for $\rho_1 \leq \rho_2$. The results in Table 6.1 reveal two interesting observations. First, we see that in all cases the expected foreground sojourn time $\mathbb{E}[S_0]$ is nearly insensitive to the job-size distribution, and only slightly increases for larger values of c_B^2. We reiterate that the results in Table 6.1 cover a wide variety of scenarios, ranging from light to heavy background loads, from symmetric to strongly asymmetric load values, and from deterministic to highly variable job-size distributions. In addition, we observe that the "exact" values based on (6.1)-(6.6) match the simulation results very closely, as they should. Tables 6.2 and 6.3 below show the results for moderately-loaded (with $\rho_0 = 0.9$) and heavily-loaded foreground traffic (with $\rho_0 = 1.8$), respectively.

Table 6.3: $\mathbb{E}[S_0]$ under heavy-loaded foreground traffic ($\rho_0 = 1.8$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.00	0.05
deterministic	0	0.05	6.52	9.55
exponential	1	0.05	6.52	9.57
Pareto-3	3	0.05	6.54	9.60
hyper-exponential	16	0.05	6.54	9.60
Pareto-2	∞	0.05	6.55	9.65
exact from (6.1)-(6.6)	1	0.05	6.52	9.58
deterministic	0	0.10	9.51	18.62
exponential	1	0.10	9.53	18.64
Pareto-3	3	0.10	9.57	18.75
hyper-exponential	16	0.10	9.57	18.78
Pareto-2	∞	0.10	9.62	18,85
exact from (6.1)-(6.6)	1	0.10	9.54	18.59

They confirm that the results for the light foreground traffic in Table 6.1 are also valid for moderate- and heavily-loaded foreground traffic. Again, we observe the $\mathbb{E}[S_0]$ is nearly insensitive to the job-size distribution, and that the CAN-model predictions from (6.1)-(6.6) are highly accurate.

To summarize, the results in Tables 6.1, 6.2 and 6.3 demonstrate that (a) $\mathbb{E}S_0$

Table 6.1: $\mathbb{E}[S_0]$ under lightly loaded foreground traffic ($\rho_0 = 0.1$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.1	0.3	0.5	0.7	0.9
deterministic	0	0.1	0.57				
exponential	1	0.1	0.57				
Pareto-3	3	0.1	0.57				
hyper-exponential	16	0.1	0.57				
Pareto-2	∞	0.1	0.57				
exact from (6.1)-(6.6)	1	0.1	0.57				
deterministic	0	0.3	0.63	0.70			
exponential	0	0.3	0.63	0.70			
Pareto-3	3	0.3	0.63	0.70			
hyper-exponential	0	0.3	0.63	0.70			
Pareto-2	∞	0.3	0.63	0.70			
exact from (6.1)-(6.6)	0	0.3	0.63	0.70			
deterministic	0	0.5	0.70	0.80	0.94		
exponential	1	0.5	0.70	0.81	0.95		
Pareto-3	3	0.5	0.71	0.81	0.95		
hyper-exponential	16	0.5	0.71	0.82	0.96		
Pareto-2	∞	0.5	0.71	0.81	0.96		
exact from (6.1)-(6.6)	1	0.5	0.70	0.81	0.95		
deterministic	0	0.7	0.80	0.95	1.16	1.53	
exponential	1	0.7	0.81	0.95	1.17	1.55	
Pareto-3	3	0.7	0.81	0.96	1.18	1.57	
hyper-exponential	16	0.7	0.82	0.97	1.20	1.59	
Pareto-2	∞	0.7	0.82	0.97	1.19	1.58	
exact from (6.1)-(6.6)	1	0.7	0.81	0.95	1.17	1.55	
deterministic	0	0.9	0.99	1.23	1.64	2.51	6.51
exponential	1	0.9	1.00	1.24	1.65	2.53	6.55
Pareto-3	3	0.9	1.00	1.24	1.66	2.55	6.60
hyper-exponential	16	0.9	1.00	1.25	1.67	2.57	6.63
Pareto-2	∞	0.9	1.00	1.25	1.67	2.57	6.67
exact from (6.1)-(6.6)	1	0.9	1.00	1.24	1.65	2.53	6.55

Table 6.2: $\mathbb{E}[S_0]$ under moderately-loaded foreground traffic ($\rho_0 = 0.9$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.1	0.3	0.5
deterministic	0	0.1	1.06		
exponential	1	0.1	1.07		
Pareto-3	3	0.1	1.07		
hyper-exponential	16	0.1	1.07		
Pareto-2	∞	0.1	1.07		
exact from (6.1)-(6.6)	1	0.1	1.07		
deterministic	0	0.3	1.29	1.69	
exponential	1	0.3	1.30	1.72	
Pareto-3	3	0.3	1.31	1.74	
hyper-exponential	16	0.3	1.32	1.76	
Pareto-2	∞	0.3	1.32	1.76	
exact from (6.1)-(6.6)	1	0.3	1.30	1.72	
deterministic	0	0.5	1.67	2.59	6.98
exponential	1	0.5	1.69	2.63	7.06
Pareto-3	3	0.5	1.72	2.68	7.18
hyper-exponential	16	0.5	1.74	2.71	7.15
Pareto-2	∞	0.5	1.74	2.73	7.45
exact from (6.1)-(6.6)	1	0.5	1.69	2.63	7.07
deterministic	0	0.7	2.52	6.46	
exponential	1	0.7	2.56	6.55	
Pareto-3	3	0.7	2.60	6.71	
hyper-exponential	16	0.7	2.63	6.70	
Pareto-2	∞	0.7	2.66	7.05	
exact from (6.1)-(6.6)	1	0.7	2.56	6.55	
deterministic	0	0.9	6.71		
exponential	1	0.9	6.76		
Pareto-3	3	0.9	6.84		
hyper-exponential	16	0.9	6.83		
Pareto-2	∞	0.9	7.04		
exact from (6.1)-(6.6)	1	0.9	6.71		

is nearly insensitive to the job-size distribution for a wide variety of parameter settings, and (b) the optimal performance can be calculated very accurately based on (6.1)-(6.6).

6.4 Score-function based splitting method

Our splitting-and-merging method is implemented between the transport layer and the application layer, and is based on the following score function (which was introduced in [43], see also Remark 6.4.1 below) that is repeatedly measured for each of the N parallel TCP-connections: for $i = 1, \ldots, N$,

$$score_i = \frac{Q_i}{G_i} + \frac{sRTT_i}{2}. \tag{6.7}$$

Here, Q represents the length in bytes of the data that has to be transmitted, and $sRTT$ is the smoothed RTT that each TCP implementation estimates. G is the smoothed throughput (which is calculated for each individual connection), that is repeatedly estimated from the following update scheme: $G_0 := 0$, and for $t = 1, 2, \ldots$,

$$G_t = \alpha \times G_{t-1} + (1 - \alpha) \times T_t, \tag{6.8}$$

where the smoothing parameter α $(0 < \alpha < 1)$ is a constant, and T_t is the measured throughput (for each individual connection) which is continuously measured every τ milliseconds. The parameter τ balances the trade-off between computational resources for calculating the score-function periodically and the responsiveness to changes in the RTT and the throughput. The score function (6.7) estimates the time of arrival of TCP segments at the receiver re-sequencing point by accounting for its major delay factors at node i: (a) the queueing delay at the sender, estimated by the ratio Q_i/G_i, and (b) the transmission delay in the network, estimated by $sRTT_i/2$; in this way, the variations of the RTTs on the networks are accounted for implicitly.

Using the iterative scheme in (6.7) and (6.8), the method simply works as follows: After calculating the score function $score_i$ for each connection $i = 1, \ldots, N$, the method assigns the packet to the connection having the lowest score, effectively choosing the connection that has the lowest estimated delay up to the point where all traffic is merged back.

Remark 6.4.1 (Score-function method)**.** The score-function method (6.7)-(6.8) was introduced earlier by Hasegawa et al. [43], called the Arrival-Time matching Load-Balancing (ATLB) method. However, our implementation of the method leads to a number of important benefits. First, the authors in [43] propose to modify the TCP protocol and to deploy the *implementation in a separate*

gateway to avoid the difficulties of replacing parts of the operating system. Instead, our implementation overcomes these difficulties by executing the scheduling functionality in application-space of a Linux operating system with the same API towards the existing applications and using the standard TCP sockets interface for the multi-path sessions. Second, as pointed out in [43] TCP throughput degradation may occur in real network environments because the receiving buffer for sorting the data segments is of limited size. As described in [58] high data rate differences between multiple networks may cause the advertized window of the faster TCP session to reach zero because the packets from the faster connection will fill the buffer and force TCP to slow-down. Although our implementation uses a limited receiving buffer, the aforementioned TCP throughput degradations do not occur. Third, the ATLB implementation [43] requires a network proxy, which raises the need for having a separate proxy for each access network. This is undesirable because the very goal of traffic splitting is to optimize performance over those access networks. Our method, instead, overcomes this problem because it does not require a network proxy, and hence can be installed both in an end-node device and in a network proxy. These observations make our method much more easy-to-deploy in real networking environments. See also Remark 6.5.1 below for comments on a comparison of our validation experiments and the ones in [43].

6.5 Experimental validation of score-function based approach

This section summarizes the results of extensive lab experiments in which two independent access networks form a CA network, which represents the most common scenario. Our aim is to demonstrate that (a) optimal traffic splitting can be closely approached using real networks with a contemporary traffic splitting solution, and (b) our model can be applied for predicting the flow-level performance of contemporary traffic splitting solutions. To this end, we have implemented the score-function method in a realistic test-lab environment. Using this implementation, the experimental results were validated against the benchmark results obtained from (6.1)-(6.6). In Section 6.5.1 we discuss the experimental setup, and in Section 6.5.2 we give an outline of the results.

6.5.1 Experimental setup

To perform lab experiments, we have re-used the setup used for the experiments in Section 2.4, consisting of two PCs that are connected by two access networks that operate similar and independent of each other. See Chapter 2 for more details on the equipment used.

For the experiments in this chapter, both multi-homed PCs run an implementation of the traffic splitting solution in addition and measure the download response times of foreground file transfers in the presence of (file transfer) background traffic. The experimental setup is depicted in Figure 6.1, where the PC

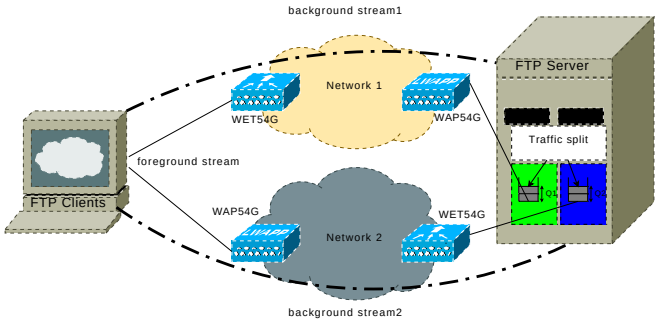


Figure 6.1: Experimental setup.

with the FTP clients generates all FTP-download requests according to independent Poisson processes, for the foreground and both background streams. It is important to state that even with a larger number of WLAN client devices there is, in addition to the access point (AP), only one station contending for the medium at the same time, as reported in [106] and observed during the experiments in [50]. Therefore, the use of only one client device yields outcomes that are representative for larger client populations.

At the PC serving as FTP server, 40,000 files have been generated according to an exponential distribution (with mean size 1×10^6 bytes) that is superimposed on a deterministic one 1×10^6 bytes to obtain a total mean file size of 2×10^6 bytes. The motivation to have at least a file size of 1×10^6 bytes is that it takes some time (and bytes) to have the splitting method operating properly and thus a bias against short files can be expected that is stronger than the one from a single TCP connection. Finally, during the experiments the files to be downloaded are randomly selected by the client PC during the experiments.

In Chapter 2 a PS-model was validated that accurately models the main aspects in file transfer response times over WLANs. Further model enhancements are presented in Section 2.2.4 to match the particularities of real network deployments, such as the presence of management traffic, different frame encapsulation and more sophisticated application behavior. Table 2.3 in Chapter 2 summarizes the parameters that are specific for our testlab networks in addition to and, where overlapping, superseding those on the IEEE 802.11b MAC in Table 2.1. In this PS-model the highly complex dynamics of the FTP/TCP/IP/MAC-stack, and their interactions, are translated into a single parameter, the effective service time. Using the effective service time, denoted β_{eff}, the effective load, denoted ρ_{eff}, is subsequently used to describe the flow-level behavior of FTP-based file transfers over WLANs without admission control as an $M/G/1$-PS model. Hence, we use the model to parameterize both our test-lab environment and the model to assess the efficiency of a TCP-based traffic split-

ting solution in the test-lab using real (WLAN) networks. In accordance with the engineering guidelines in Chapter 2 we have configured our test-lab environment similarly with the same TCP window size, a sufficiently large AP buffer to avoid overflows, and restricted the maximum load per network (to values below $\rho_{eff} = 0.9$). In these circumstances, it was shown that the model leads to highly accurate predictions over a wide range of parameters combinations, including light- and non-light-tailed file-size distributions and light- and heavy-load scenarios. In addition, the observed mean download response times are in these circumstances fairly insensitive to the file-size distribution, as suggested by the PS-model.

Another important aspect is how similar both access networks will perform. In the experimental validation in Chapter 2 the same networks are used as in the experiments described in this chapter. The validation results of both networks are shown in Section 2.4.2 and indicate that both networks perform very similar. The only noticeable difference between both networks is that a different channel frequency is configured.

Based on the parameters in Tables 2.1 and 2.3 from Chapter 2, we obtain from the model an effective service time, $\beta_{eff_{bg}}$, of 3.041 seconds of the background file transfers in network one and two. A slightly higher value of 3.058 seconds applies to the foreground file transfers, called $\beta_{eff_{fg}}$. This difference is explained by the eight bytes of overhead that is introduced by our implementation of the splitting solution to transfer the foreground traffic.
To make the adjustment to changing network conditions very responsive, in our experiments the smoothing parameter α of the score-function, presented in (6.8) was set to 0.4. Hasegawa et al. [43] argue that the measurement interval τ should be fixed and set to a value of 100 ms for network with an RTT of 20 ms. However, test-lab experimentation (results are not presented here) shows that it is better to adapt τ to changing RTT values in the network. More precisely, a best practice rule-of-thumb is to set τ equal to $4 \times RTT$ because a shorter period does not give a good performance estimation (due to the impact of fast retransmissions and task scheduling effects of the PC's operating system).

While in operation, the splitting solution records the number of bytes over each network for each of the file downloads processed. For an experiment in which K foreground file transfers are processed by the splitting solution and where the number of bytes for file transfer k over network i is denoted B_{ki}, the average file splitting ratio over network i denoted $\hat{\alpha}_i^{exp}$, is defined by:

$$\hat{\alpha}_i^{exp} := \frac{1}{K} \sum_{k=1}^{K} \frac{B_{ki}}{\sum_{n=1}^{N} B_{kn}}, \tag{6.9}$$

where $N = 2$ for our test-lab environment.

6.5.2 Experimental results

In this section we report on the experiments conducted in our test-lab environment to determine the efficiency of the TCP-based splitting solution in a real networking environment. The efficiency is determined by comparing the mean file download times in our test-lab environment against the predictions from the dynamic split model.

This model for predicting the file download times consists of two components. The first component is the WLAN PS-model from Chapter 2 that is able take the WLAN specific parameters into account to determine the effective service time of the file transfers and the effective load of the network to obtain the expected download time in one PS node. The second component of the model considers ideal dynamic job-splitting of foreground traffic streams over $N = 2$ PS-queues. In Section 6.3 it is shown that the results obtained from numerically solving the continuous-time Markov chain from Section 6.2 match very closely with the ones from the simulation program. Therefore either the simulation program of the Markov chain may be used to determine the performance of ideal dynamic traffic splitting. For comparing the model-based results of dynamic traffic splitting in real networks, the simulation program introduced in Section 6.3 is used rather than the Markov model from Section 6.2. The reason for using the simulation program is that exactly the same service time distribution is simulated as the file size distribution in our experiments. Moreover, the simulation program calculates, in addition to the download response times of the foreground and background traffic streams, the average job-splitting ratio, denoted $\hat{\alpha}^*$. The average job-splitting ratio in our simulation program is calculated by (6.9) and replacing the number of bytes B_{ki} that traversed for download k network i by the amount of service β_{ki} that job k received in PS node i.

The simulations have been conducted for the same job-size distribution as the file-size distribution used in our experiments, which is an exponential distribution with mean $\beta/2$ superimposed on a deterministic value of $\beta/2$. In the simulations, the mean service time of foreground and background jobs, β, was set to unity. To determine representative values from the simulations, the mean sojourn time values of the foreground stream, denoted $\mathbb{E}[S_0|\beta = 1]$, and background streams in network one and two, $\mathbb{E}[S_i|\beta = 1]$ for $i = 1, 2$, are used as follows:

$$\mathbb{E}[S_0|\text{model}] = \mathbb{E}[S_0|\beta = 1]\,\beta_{eff_{fg}}, \tag{6.10}$$

$$\mathbb{E}[S_i|\text{model}] = \mathbb{E}[S_i|\beta = 1]\,\beta_{eff_{bg}} \quad (i = 1, 2), \tag{6.11}$$

to obtain the values that are predicted for optimal dynamic splitting in the test-lab networks for the given effective load values of the foreground and background file downloads. Exactly the same effective-load values are used for both

the simulations and the experiments.

For our experiments we have set the effective load for the foreground traffic and the background traffic in network one and two, ρ_0, ρ_1, ρ_2 respectively, and determined the download request arrival rates for the FTP clients using the effective service time model from Section 2.2.4 In Table 6.4 the runs that we have performed are shown with their approximate effective load values. To obtain sufficiently small 95% Confidence Intervals (CI) for the measured average download response times, the experiments required an execution time from 2-10 days per run. As a result, the 95% CI is typically in the range of $1-2\%$ with a maximum of 4% for the foreground mean download response times $\mathbb{E}[S_0|exp]$ and typically in the range of $2-4\%$ with a maximum of 5% for the background mean download response times in network one, $\mathbb{E}[S_1|exp]$, and two, $\mathbb{E}[S_2|exp]$.

Three columns in Table 6.4 are labeled $\mathbb{E}[S_i]$ for $i = 0, 1, \ldots, N$ that represent the mean file download times of stream i. In each of these columns, a triple $(\mathbb{E}[S_i|\text{exp}], \mathbb{E}[S_i|\text{model}], \Delta\%)$ is shown for stream i, where the first element is the measured average file download response time, the second element the model prediction of ideal download performance and $\Delta\%$ as the relative difference between both elements that is calculated as follows:

$$\Delta\% = \frac{\mathbb{E}[S_i|\text{exp}] - \mathbb{E}[S_i|\text{model}]}{\mathbb{E}[S_i|\text{model}]} \times 100\%. \tag{6.12}$$

The column labeled α shows the couple $(\hat{\alpha}^{exp}, \hat{\alpha}_i^{model})$, representing the average splitting ratio obtained from the experiments and from the model respectively.

ρ_0	ρ_1	ρ_2	$\mathbb{E}[S_0]$	α	$\mathbb{E}[S_1]$	$\mathbb{E}[S_2]$
0.1	0.1	0.5	(2.268, 2.145, 5.7%)	(0.610, 0.602)	(3.614, 3.575, 1.1%)	(6.622, 6.422, 3.1%)
0.1	0.2	0.3	(2.051, 2.029, 1.1%)	(0.524, 0.524)	(4.049, 4.006, 1.1%)	(4.561, 4.578, -0.4%)
0.1	0.3	0.7	(3.008, 2.914, 3.2%)	(0.647, 0.643)	(4.763, 4.714, 1.0%)	(11.061, 11.036, 0.2%)
0.1	0.4	0.5	(2.693, 2.646, 1.8%)	(0.532, 0.532)	(5.437, 5.438, 0.0%)	(6.435, 6.524, -1.4%)
0.1	0.4	0.6	(3.053, 2.913, 4.8%)	(0.577, 0.570)	(5.535, 5.481, 1.0%)	(8.484, 8.216, 3.3%)
0.1	0.5	0.5	(3.043, 2.881, 5.6%)	(0.500, 0.500)	(6.619, 6.569, 0.8%)	(6.676, 6.568, 1.6%)
0.5	0	0.8	(3.965, 3.787, 4.7%)	(0.787, 0.775)	(0.000, 0.000, 0.0%)	(23.007, 23.721, -3.0%)
0.5	0.1	0.2	(2.559, 2.419, 5.8%)	(0.525, 0.522)	(4.496, 4.439, 1.3%)	(5.101, 4.979, 2.5%)
0.5	0.2	0.3	(2.932, 2.788, 5.2%)	(0.526, 0.525)	(5.330, 5.219, 2.1%)	(5.985, 5.946, 0.7%)
0.5	0.3	0.7	(5.259, 4.922, 6.9%)	(0.662, 0.658)	(7.809, 7.539, 3.6%)	(17.103, 17.297, -1.1%)
0.5	0.5	0.5	(4.994, 4.900, 2.0%)	(0.500, 0.500)	(10.529, 10.399, 1.3%)	(10.620, 10.400, 2.1%)
0.5	0.5	0.6	(6.069, 5.975, 1.6%)	(0.544, 0.544)	(11.734, 11.537, 1.7%)	(14.254, 14.369, -0.8%)
0.9	0.2	0.3	(4.782, 4.526, 5.7%)	(0.530, 0.527)	(8.194, 8.154, 0.5%)	(9.637, 9.241, 4.3%)
0.9	0.3	0.4	(6.589, 6.426, 2.5%)	(0.532, 0.532)	(11.866, 11.759. 0.9%)	(13.565, 13.598, -0.3%)
1.2	0	0	(3.952, 3.870, 2.1%)	(0.499, 0.500)	-	-

Table 6.4: Measurement results testbed.

Based on the outcomes in Table 6.4, it can be concluded that nearly optimal performance can be achieved by our score-function based algorithm for routing TCP segments. The differences between the benchmark and the measured performance are very small, and typically no more than a few percent. Considering

the foreground traffic splitting ratios, the solution has the tendency to slightly overload the network with the smallest average background load. Overall, the foreground download response times show that non-ideal traffic splitting on a packet-basis may deliver near optimal foreground traffic performance.

One remarkable run that exemplifies the good performance of dynamic traffic splitting is the one in the third row, where $\rho_0 = 0.1$, $\rho_1 = 0.3$ and $\rho_2 = 0.7$. Here the measured average download response time is even lower than its effective service time of 3.057 seconds, even in the presence of an effective load of 1.1 on a total system capacity of 2.

Given the limited sensitivity of the sojourn time to the job-size distribution, it can be expected that the test-lab results are also representative for other file-size distributions, as long as a minimum size is respected that is comparable to the one we used in our experiments. For experiments in which one of the networks was heavily loaded ρ_1, $\rho_2 \geq 0.8$, the differences are getting large. This may be explained by the higher number of capacity fluctuations that may be perceived by the TCP sessions to which the traffic distribution may need to respond.

Remark 6.5.1 (Efficiency)**.** Hasegawa et al. [43] also report valuable but quite limited experimental results on the score-function method. However, there are some important differences. First, in our experiments the background traffic is assumed to be highly dynamic. In our experiments background jobs arrive according to Poisson processes, and the the job-size distributions may be highly variable, whereas the background streams in [43] are persistent and hence do not require a highly adaptable traffic-splitting algorithm. Second, the experiments in [43] are focused on capacity aggregation and on optimizing throughput over parallel networks. In contrast, our analysis and experiments are oriented towards minimizing expected response times. Moreover, our measurements allow us to quantify the efficiency of our splitting algorithm by comparing the results with the benchmark results from our analytical model.

6.6 Comparison between Concurrent Access strategies

In this section we compare the outcomes of five CA strategies in a network with $N = 2$ Processor Sharing queues. The first strategy statically assigns foreground jobs to the one most suitable queue based on a long-term characteristic. Although this is a trivial policy, it is commonly used in practice and ignores the available resources in the other queue. The static assignment policy in our comparison assigns all foreground traffic to the queue that has the lowest average background traffic load. Note that the expected sojourn time obtained for static assignment can be calculated in a straightforward manner. The second strategy is the static split strategy outlined in Chapters 3 and 4, where each of the foreground jobs is split into N fragments according to the optimal static splitting rule, $\underline{\alpha}^*$, obtained from the simulations in Chapter 4 that minimizes

the expected transfer time of foreground jobs that are split over N parallel PS-queues. The third strategy in the comparison is the Bayesian learning algorithm presented in Chapter 5 that splits a stream of jobs by optimally assigning entire jobs to different queues, such that the expected sojourn time is minimized. Because this algorithm uses the occupation level in each queue, we consider this Bayesian algorithm a dynamic job-assignment approach. These strategies are compared against the dynamic job-split approach presented in this chapter. To benchmark these strategies against one that is well-known in the literature, simulation results on the Join-the-Shortest-Queue (JSQ) approach are included.

In the following subsections the results of calculations (only in the case of static assignment) and simulations (for all other strategies) are shown in figures. In Appendices B-D, the same results can be found in tables. Motivated by the full or near-sensitivity of the mean foreground sojourn time to the job-size distribution for the static assignment, static splitting and dynamic splitting policies, Poisson arrivals with an exponential job-size distribution are used, where applicable, in our comparison. We assume a network, where $N = 2$ and $\beta^{(1)} = 1$. The load of the foreground traffic ρ_0 was varied from light ($\rho_0 = 0.1$) to mild ($\rho_0 = 0.5$), moderate ($\rho_0 = 0.9$) and heavy ($\rho_0 = 1.8$), and the background loads ρ_1 and ρ_2 were varied as $0.1, 0.2, \ldots, 0.9$. Each simulation run is based on averages from 10^8 foreground observations.

Our main interest is in the traffic splitting or assignment ratio of the foreground traffic and in the mean sojourn times of the foreground and background traffic, both as a function of the traffic load and the distribution strategy. The background traffic performance in both queues is consolidated into the following notion of the background traffic performance:

$$\gamma = \frac{\mathbb{E}[S_1]\,\rho_1 + \mathbb{E}[S_2]\,\rho_2}{\rho_1 + \rho_2} \tag{6.13}$$

6.6.1 Light foreground traffic load

Figure 6.2 shows that under light foreground traffic load, the traffic splitting/assignment ratios demonstrate very typical behavior for each strategy. The static assignment strategy simply selects the queue with the smallest background traffic load and is a trivial example. When performing static job-splitting, the ratio remains close to half when the background traffic loads become unequal and cause a very steep increase in the splitting ratio towards a highly unbalanced system. The ratio in which the jobs are assigned by the Join-the-Shortest Queue (JSQ) algorithm differ much from the other in that there seems to be a linear increase as the systems become more and more unequally loaded. Different from the other strategies, both the dynamic Bayesian assignment and the dynamic splitting policies show a bended curve as the queues become more and more unequally loaded. However, the dynamic splitting ratio shows a much steeper increase and remains far below the other strategies.

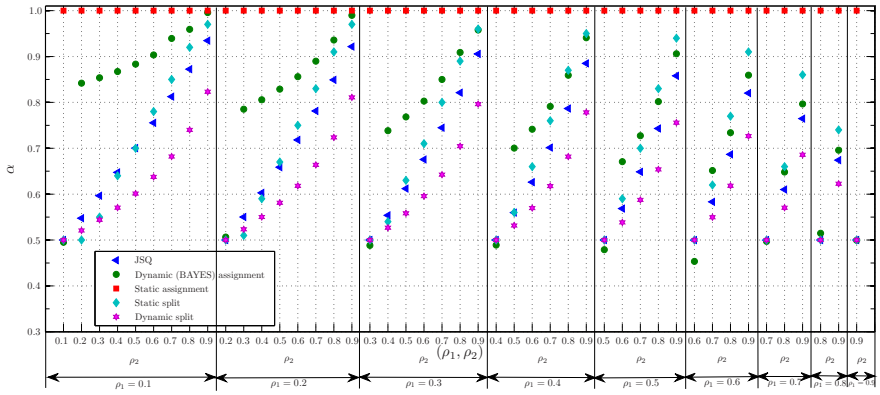


Figure 6.2: Splitting/assignment ratios of light foreground traffic ($\rho_0 = 0.1$) for various CA strategies as a function of the background load.

The sojourn times that match the splitting/assignment ratios from Figure 6.2 are shown in Figure 6.3. From the perspective of the foreground sojourn time, the dynamic traffic splitting outperforms all other strategies. As long as the overall background traffic load is not very high, static assignment performs fairly well considering its simplicity. If the background load in both queues increases, static assignment may yield very high sojourn times or may cause one of the queues to become unstable where other strategies continue to perform well.

Static splitting of light foreground traffic leads to fairly good performance as long as both queues are lightly loaded. As the background load on one of the queues is gradually increased, the performance of static splitting is very similar to static assignment for low to moderate overall background traffic loads. The results for Bayesian assignment and JSQ are so close that they can hardly be distinguished. In particular under higher overall background traffic loads is the performance of dynamic assignment and JSQ leading to fairly good results. Clearly, there is no strategy that outperforms dynamic splitting. The relative difference to static splitting is not very large under very low overall background traffic loads. If, however, the background traffic loads increase the difference between dynamic splitting and static splitting becomes very large, up to the point where dynamic splitting is more than twice as fast.

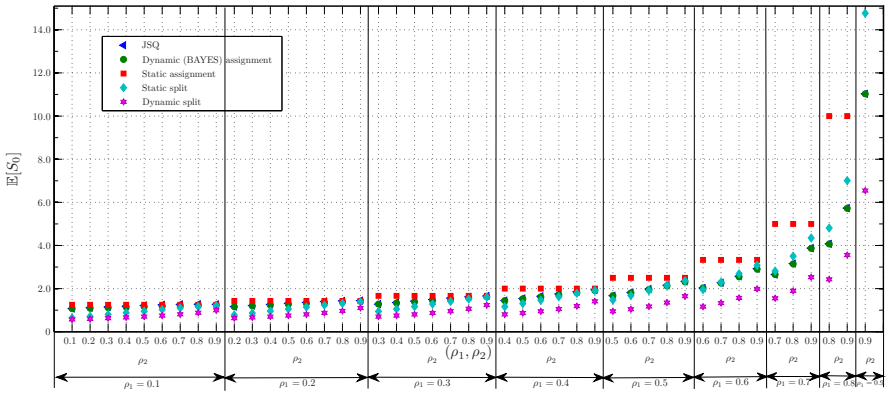


Figure 6.3: Mean sojourn times of light foreground traffic ($\rho_0 = 0.1$) for various CA strategies as a function of the background load.

The impact of the traffic distribution strategies on the background traffic performance is shown in Figure 6.4. Here it is shown that the relative differences between the background traffic load is limited as long as the queues are not very unequally loaded with background traffic and the overall background traffic load is not too high. The overall background traffic sojourn times is most favorable in the remaining cases for either static assignment low to medium background traffic load) or in the case of dynamic assignment or JSQ. Under very high background traffic load, dynamic splitting improves the foreground traffic performance slightly at the expensive of the background traffic compared to JSQ and Bayesian assignment. When considering the splitting ratios in Figure 6.2, dynamic splitting is directing much more service demand to the queue with the highest load. This consistent behavior can be observed throughout Figure 6.2. Static splitting performs worst at very high load, both with respect to the foreground-and the background performance. This may be explained by the high fluctuations in occupation level in both queues: splitting the foreground traffic based on a long-term characteristic does not match the occupation levels at small time scales.

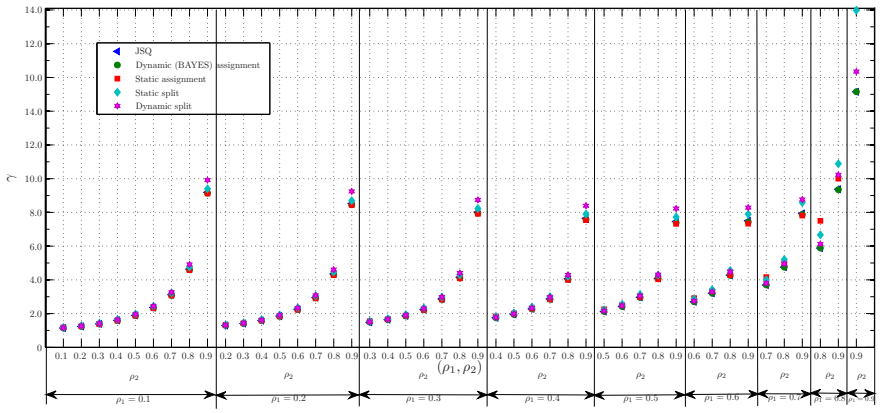


Figure 6.4: Background traffic performance (γ) for various CA strategies with light foreground traffic ($\rho_0 = 0.1$) as a function of the background load.

6.6.2 Mild foreground traffic load

Figure 6.5 shows the results on the splitting/assignment ratios for a foreground traffic load that is equal to $\rho_0 = 0.5$. As expected, the static assignment strategy has fewer combinations for which the system is stable. Considering the curves of the other CA strategies it may seem to increase steeper than for the light background traffic load. This is not generally the case. For static splitting, the ratios are increasing much slower with an increasing background load skewness when compared to light foreground load. This effect is even more pronounced for Bayesian assignment.

The JSQ strategy increases its assignment ratios very modestly. An exception to this rule is the behavior of dynamic splitting, which is splitting the traffic such that, in comparison to light foreground load, a higher portion of the foreground traffic arrives at the queue that has a higher load. Due to the higher foreground traffic load, the splitting ratios show smaller variations, in particular for higher background load values. Again are the dynamic policies characterized by bended curves whereas the JSQ strategy demonstrates an almost linear increase. The impact on the foreground sojourn times is illustrated in Figure 6.6, where the static assignment strategy is clearly performing much worse than others. Also the distance between these other strategies has increased in comparison to the case of light foreground traffic load (in Figure 6.3). Figure 6.6 also shows that for mild foreground traffic load the difference between the JSQ strategy and dynamic Bayesian assignment becomes visible for higher background load values. For the highest background load depicted, the results are very similar to the ones illustrated in Figure 6.3. The results on the background performance are shown in Figure 6.7 and demonstrate higher differences between the various

strategies. It can be observed that static assignment of foreground traffic does not lead to poor background performance.

In fact, for low overall background traffic loads, static assignment leads to the lowest background sojourn times. Dynamic assignment using JSQ or Bayesian strategies leads to slightly higher sojourn times than those for static assignment. For higher overall background loads the difference between Bayesian assignment and JSQ becomes visible and clearly Bayesian assignment delivers lower foreground response times at the expense of the background traffic performance. Dynamic splitting leads to relatively high background sojourn times when both queues are exposed to very different background traffic loads. In those cases dynamic splitting delivers higher background sojourn times than static splitting. The latter strategy adversely affects the background performance most prominently when the overall background traffic load is high, an effect that is similar to the foreground sojourn times in Figure 6.6.

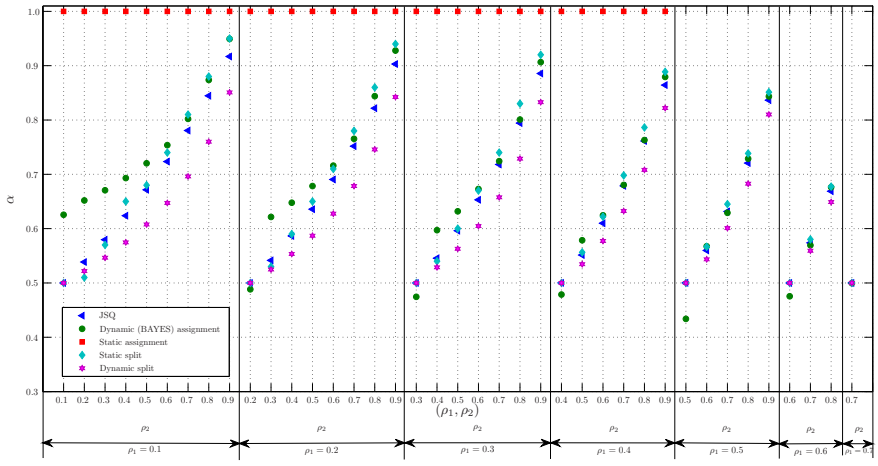


Figure 6.5: Splitting/assignment ratios of mild foreground traffic ($\rho_0 = 0.5$) for various CA strategies as a function of the background load.

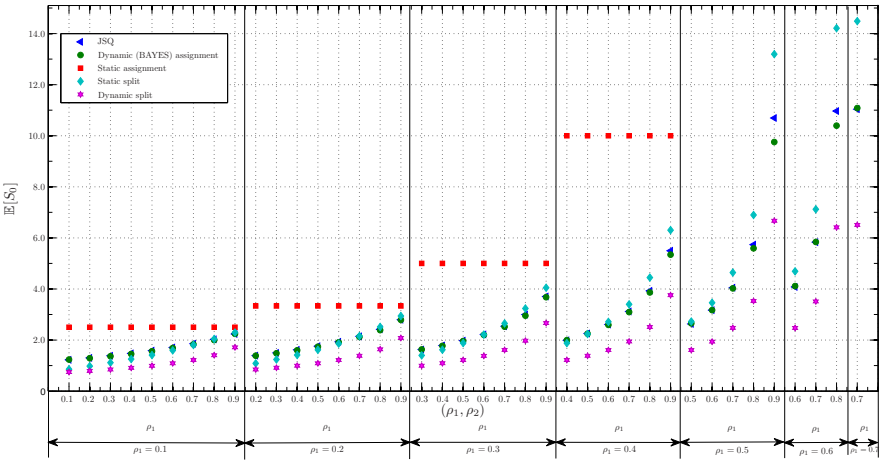


Figure 6.6: Mean sojourn times of mild foreground traffic ($\rho_0 = 0.5$) for various CA strategies as a function of the background load.

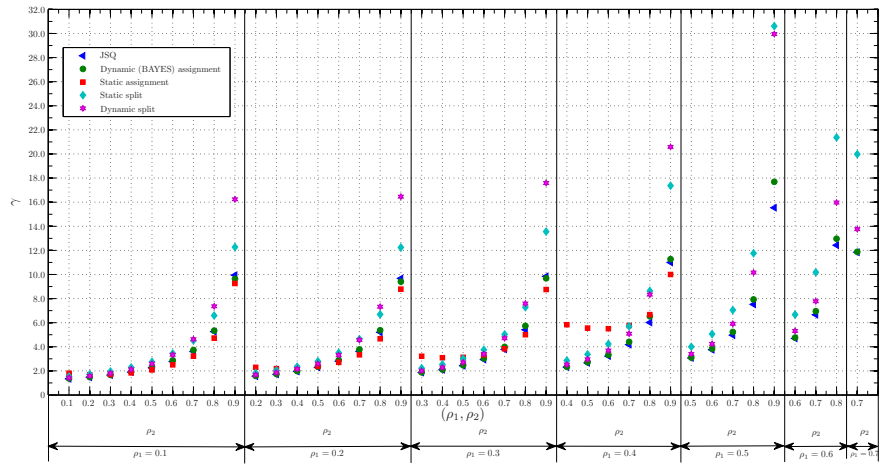


Figure 6.7: Background traffic performance (γ) for various CA strategies with mild foreground traffic ($\rho_0 = 0.5$) as a function of the background load.

6.6.3 Moderate foreground traffic load

For a moderate foreground traffic load, $\rho_0 = 0.9$ the splitting/assignment ratios are as expected: the differences between the curves are again smaller in compari-

son to lower foreground traffic loads and the splitting ratios of dynamic splitting are lagging behind the other strategies. In the case of moderate foreground traffic load static assignment does not lead to a stable system in the case of having a background load equal to 0.1. The differences between static splitting and dynamic assignment using JSQ or Bayes are very small. Similar to what was observed for an increasing foreground traffic load from light to mild intensities, dynamic splitting is the only strategy that is directing again more traffic towards the queue that has the highest background load. In contrast, Bayesian assignment and static splitting send a smaller portion of the foreground traffic to the queue with the highest background load.

The JSQ strategy has the smallest sensitivity to the foreground traffic intensity; slightly smaller amounts of jobs are assigned to the queue with the highest background load. The resulting impact on the foreground sojourn times are shown in Figure 6.9 where it is clearly shown that Bayesian assignment outperforms the JSQ strategy most notably for unbalanced systems. Again, the sojourn times of the foreground traffic are significantly smaller than for all other strategies. The resulting background traffic sojourn times demonstrate that dynamic splitting optimizes the foreground traffic at the expense of the background performance, in particular background jobs in highly loaded queues in unbalanced systems experience very high mean sojourn times.

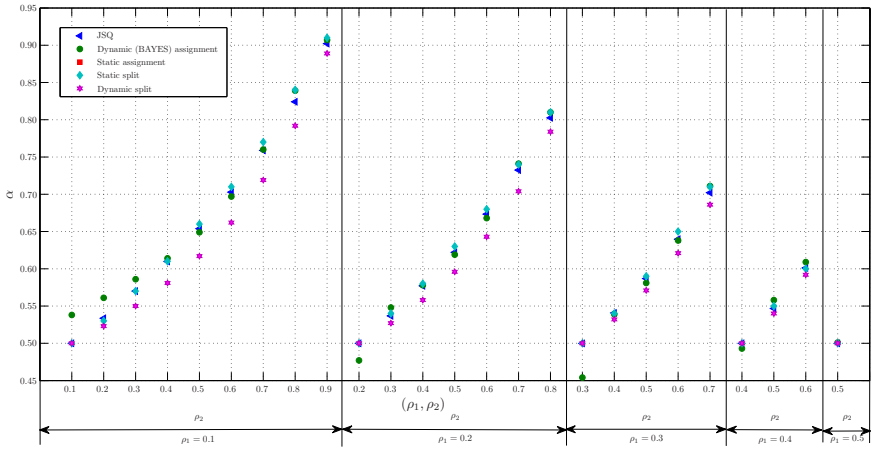


Figure 6.8: Splitting/assignment ratios of moderate foreground traffic ($\rho_0 = 0.9$) for various CA strategies as a function of the background load.

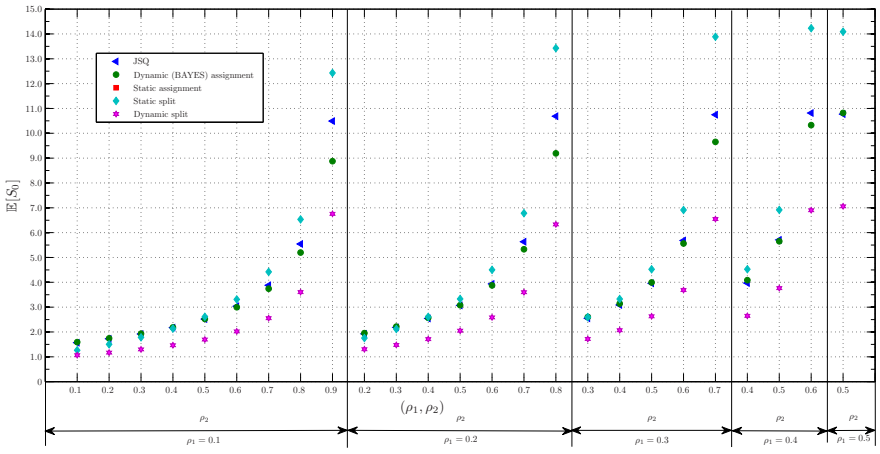


Figure 6.9: Mean sojourn times of moderate foreground traffic ($\rho_0 = 0.9$) for various CA strategies as a function of the background load.

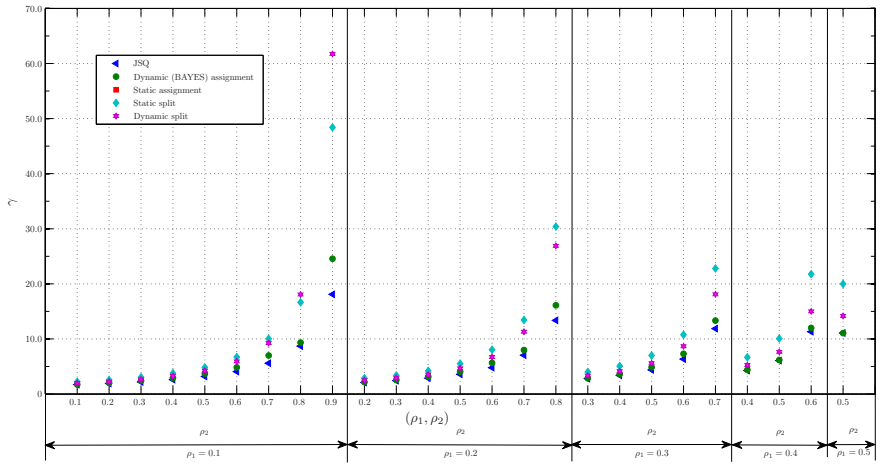


Figure 6.10: Background traffic performance (γ) for various CA strategies with moderate foreground traffic ($\rho_0 = 0.9$) as a function of the background load.

6.6.4 Heavy foreground traffic load

When exposing the $N = 2$ queueing network to a high foreground traffic load of $\rho_0 = 1.8$ the four strategies shown in Table 6.5 may lead to a stable system. Dynamic splitting sends the most traffic over the queue that also carries back-

ground traffic. Bayesian assignment performs slightly better with respect to foreground traffic performance than JSQ. In this context the relative difference between static splitting and the other approaches is fairly limited when considering its simplicity. The foreground traffic performance of dynamic splitting is 8.3% better than the second-best strategy of Bayesian assignment. When taking an overall performance metric $\psi = \rho_0 \cdot \mathbb{E}[S_0] + \rho_2 \cdot \gamma$ dynamic splitting also performs best, improving the performance with 4.9% in comparison to JSQ.

	α^*	$\mathbb{E}[S_0^{\alpha^*}]$	γ
Static split	0.527	11.723	20.555
Dynamic (Bayes) assignment	0.533	10.370	15.391
Dynamic split	0.525	9.510	19.207
JSQ	0.527	10.532	10.682

Table 6.5: Simulation results for heavy foreground load ($\rho_0 = 1.8$), with $\rho_1 = 0$ and $\rho_2 = 0.1$.

6.7 Conclusions and further research

In this chapter the dynamic splitting of foreground traffic jobs over multiple PS-queues is considered. An analytical model is described that allows to determine the "optimal" traffic splitting performance by solving a CMTC. This analytical model is validated by simulations for various types of service time distributions and traffic intensities. It is shown that this "optimal" performance can be closely approached in a real network deployment where the networks can be considered to behave similar to PS-queues, according to the effective-service time model described in Chapter 2. Experiments reveal close to optimal performance can be achieved by using a remarkably simple and easy-to-deploy score-function based algorithm that dynamically splits TCP segments based on on-the-fly measurements on the per-node RTT, the sending-buffer contents and the estimated throughput. We believe that this forms a significant step towards filling the gap between the theoretical modeling and analysis of traffic splitting among multiple network paths and practice.

In comparison to other traffic splitting and assignment strategies that are presented in the previous Chapters the foreground traffic performance delivered by dynamic splitting is better than that of the other strategies. It is demonstrated that dynamic splitting is sending relatively larger portions of the foreground traffic to the queue with the highest average load. In fact, when the foreground traffic intensity increases, these portions become even larger as opposed to the other strategies. By doing so, dynamic splitting consumes all available resources on every queue at the expense of the background traffic. This can be explained by the fact that dynamic splitting operates at an infinitely small time scale, as opposed to the other strategies that operate on the time scale of processing an

entire job or its halves. Unlike the other strategies, perfect dynamic splitting benefits from short-term under-utilizations, no matter how small in length or magnitude.

The results presented in this paper lead to a variety of challenges for follow-up research. First, as pointed out in Chapter 3, the accuracy of PS models becomes questionable under high traffic loads or system under-dimensioning so that (bursty) packet loss has a significant impact on the results. It needs further exploration how the parameterizations in (6.7) and (6.8) need to be adjusted, e.g. by *adapting* the smoothing parameter α and the measurement interval τ, in response to bursty packet loss patterns.

Second, the CAN-model is based on the assumption that the traffic is elastic, in the sense that the available bandwidth changes over time caused by fluctuations in the number of simultaneous flows in the network. However many applications (such as streaming applications) are not elastic, and instead require fixed amounts of network bandwidth. To incorporate smart traffic splitting for mixtures of elastic and streaming applications, the model should be extended to mixtures of fixed and elastic traffic streams. Optimal splitting in such multiclass models opens up a challenging area for follow-up research.

Third, the fluid-based dynamic splitting of foreground jobs (as discussed in Section 6.2) has the attractive property that the synchronization delay is zero, while at the same time the policy is fair in the sense that foreground jobs receive no more than their fair share of the capacity at each of the individual nodes. It seems obvious that even better performance for the foreground traffic can be obtained by dropping the restriction of PS-based fair sharing of capacity for each node, and allowing for the possibility of some kind of weighted sharing of the capacity of the nodes in favor of the foreground traffic, which clearly comes at the expense of the background traffic streams. Further optimization of the performance of foreground streams, and balancing the trade-off with the degradation of the performance of the background streams opens up an interesting direction for further research.

Fourth, the experimental validation is conducted in a lab-environment with two similar access networks. However, in practical circumstances heterogeneous access networks may be connected by a Wide Area Network (WAN). This network heterogeneity may result in very different path delay and packet loss characteristics for each connection, whereas the WAN network may merge all connections and impose a shared bottleneck. To this end, further experiments should reveal the behavior of the splitting solution for a broader set of conditions that may occur in practice.

Fifth, when comparing the CA strategies it was observed in the simulation results that the assignment ratio of the foreground traffic for the Bayesian dynamic strategy is not always 0.5 for two equally loaded PS-queues, despite the

fact that the mean sojourn times are according to expectations. It needs to be investigated why the observed assignment ratios for the Bayesian assignment strategy do not always meet the expected values.

Finally, another topic of further research is to study the impact of the arrival process on the download response times, if for instance an MMPP arrival process is considered instead of Poisson.

Chapter 7

Traffic Control in Wireless Networks

In Chapter 2 we proposed a concept for modeling complex combined dynamics and overhead of multiple protocol layers from a wireless network into an explicit expression for the effective service time of jobs in a PS-queue. In this way, files that are transferred through a network can be modeled by jobs that are processed in a PS-queue, which allows accurate predictions of the mean download times of file transfers using a very simple model.

Another application of the PS model with the notion of effective service time is found in Chapters 5 and 6 to predict the performance of file transfers that are respectively assigned to or split over multiple wireless networks.

In this chapter a traffic control solution is proposed that relies on the basics of the performance model from Chapter 2 for realizing Quality of Service (QoS) guarantees of wireless users in a network where the available capacity may fluctuate. It is shown that the throughput parameter bits/s does not provide sufficient insight in load conditions and/or traffic demands in wireless networks.

Therefore, we define a multi-service traffic profile that does provide this insight, which can be used to define the notion of a QoS budget. The QoS budget is assigned to a terminal in accordance with the method described in [47]. Using this method, the terminal can determine locally if the network consumption does not exceed the assigned budget and whether a new application session may fit within the QoS budget given. Finally, it is shown how the QoS budget with its multi-service traffic profiles can be used as a dynamic solution to guarantee the QoS of various applications in a wireless network, where channel conditions may vary over time and stations may move around.

This chapter is based on the results presented in [93] and [94].

7.1 Introduction

A wide range of wireless networks exists that provide mobile users access to the Internet. In order to satisfy the user's QoS expectations, resource allocation mechanisms and QoS guarantees should exist that can also be found in cellular networks. Some wireless technologies, however, do not use adequate resource allocation mechanisms and therefore guarantees cannot be given.

A QoS enabled network distinguishes various priority classes, reflected in the packet header, such that the network can differentiate in packet serving times. Each priority class may differ in the QoS targets/guarantees and the various packet scheduling mechanisms (i.e., WFQ, DRR) that aim at realizing these targets. Standardization in 3GPP, 3GPP2, IEEE 802.1Q, IEEE 802.16 and the wireless multimedia extensions of WLAN (subset of the IEEE 802.11e standard) tends to distinguish up to four different traffic classes: the conversational, streaming, interactive and the background class (using UMTS terminology). If, however, the traffic ingress volume of a network exceeds the egress limit, these mechanisms are not sufficient and degradation of quality is inevitable. This is why - besides priority - QoS solutions must include indications about volumes, specified in a traffic contract. The traffic contract dictates how much traffic users can offer to a network, which may depend on the network load, time of day, day of month, etc. In addition to time-varying user demands there may be a second source of dynamics involved in contemporary wireless networks: time-varying connection speeds due to transmission rate adaptation algorithms.

A prominent example of a network that employs transmission rate adaptation is WLAN: if the signal quality between a WLAN station and its AP declines, the transmission rate of a station may decrease as a result of rate adaptation and consequently the WLAN overhead increases. Hence, the length of a bit on the WLAN medium differs per station which makes the traditional parameter bit/s an inaccurate measure for specifying application demands. As a result, an effective alternative for the traffic contract is required. The variation of the length of bits traversing this medium also affects the time scale where the mechanism operates that maintains the desired QoS level for end-users. This mechanism should not only assign capacity on a long time scale (as happens with admission control) but also needs to re-distribute this capacity over associated stations on a relatively short time scale, if channel conditions demand so.

The solution proposed in this chapter is aimed at solving user continuation control, rather than only admission control. This is realized by taking into account the dynamically changing channel conditions in the (re)assignment of airtime capacity. The result is a single formula that can be applied to maintain the desired QoS level on two time scales:

1) the minutes time scale, where the decision has to be made whether a newly associated station can be added to the system, and
2) the seconds time scale, because airtime capacity needs to be re-distributed among a group of stations if their connection speeds changed as the result of e.g., movements.

Different prioritization policies can be applied. By giving priority to stations with high connection speeds the overall capacity is maximized. End-user priority is achieved by calculating the needed resources based on the desired throughput at the application-layer for those users that are considered most important. Finally, a fair sharing policy can be applied by distributing the available airtime capacity equally over all associated stations. Although the solution is made suitable for WLAN networks, the approach of using multi-service traffic profiles to realize and maintain QoS guarantees is also applicable to other rate-adapting networks.

Among the various QoS challenges, admission control is an important component for the provisioning of guaranteed QoS parameters. The purpose of admission control is to limit traffic or users so that the QoS of existing users or flows will not be affected while the medium resources can be maximally utilized and attracted the attention of many researchers [126], [40], [8], [39],[98],[90],[88], of which [39] provides a good overall overview. Two main categories of admission control approaches are distinguished, namely measurement-based and calculation-based. In general, the measurement-based approaches form a good basis to realize user continuation as they provide access to frequently changing medium conditions.

In the area of measurement-based approaches Distribution Admission Control (DAC) was proposed in [121, 122] by the IEEE 802.11e working group. It measures the channel occupancy for each of the priority classes used and hence could in principle be extended with dynamic user continuation. However, the DAC approach does not provide a direct relation between the MAC parameters for allocating transmission time and QoS requirements from applications. The virtual MAC, as introduced in [14, 118], is an elegant solution to be executed on the mobile hosts for scheduling virtual packets on a virtual medium to decide on whether new flows can be admitted. The major drawback of this approach is that a lot of processing power is consumed on the involved wireless devices, further reducing their limited battery life.

Another measurement-based approach is threshold-based admission control in

which flows are admitted to the medium, depending on the average load or collision rate. Additional criteria that take into account user continuation could have been added. Although the implementation remains rather simple, the threshold values need to be determined and form no guarantee to the instantaneous QoS metrics. In the Harmonica approach the decision to admit new flows is based on their throughput requirement and is made at the expense of best-effort traffic and dynamically adjusted IEEE 802.11e channel access parameters [123], which still is a challenging problem.

In the context of calculation-based admission control schemes, Markov chain models (e.g. from Bianchi [21]) are often proposed, e.g. by [98, 13], to calculate the expected throughput for a certain amount of stations and traffic demand. There are two problems with applying these Markov models: first, they assume a saturated medium, and second, they operate on a long time scale and hence are not effective to deal with the properties of the WLAN channel that can vary on a millisecond time scale.

It should be pointed out that in the above-mentioned approaches and those explained in [40, 85] the transmission rate of the involved stations is not accounted for, in spite of its major influence on throughput performance. The throughput performance for all users is affected in part because bits become longer, and raises the medium occupation level, but possibly also due to the use of lower transmission rates for control traffic and growing MAC overhead as a result of backward compatibility with older standards. The only known approach that does take this effect into account is presented by Gao in [39] as physical-rate-based admission control, but this approach relies on the polling-based HCCA model, that is, similar to the legacy PCF polling function, hardly used in practical deployments.

This chapter is structured as follows. Section 7.2 shows why throughput parameter bits/s does not provide accurate insight in load conditions and/or traffic demands in WLANs. It proposes a multi-service traffic profile to overcome this aspect and then shows how this profile can be used to compute the WLAN airtime consumption, a parameter that is subsequently used in Section 7.3 for both the user continuation control as well as for the proper distribution of WLAN capacity over all stations. The multi-service traffic profile is natural for real-time applications; simulation results are used to compute the multi-service profile parameters for elastic applications. Section 7.3 discusses the relation of the proposed solution to the admission control approaches reported in literature. Section 7.4 shows the lessons learned from implementing the proposed solution.

7.2 Multi-service traffic profiles

The Service Level Agreement (SLA) of traffic contracts traditionally consists of committed/peak data rates and maximum burst sizes. If a SLA concerns

the amount of traffic at the IP layer, the capacity demand on lower protocol layers should be known to make any form of guarantee. This section shows that SLAs for WLANs are not sufficiently defined with the parameter bits/s. A traffic profile is proposed that consists of two parameters to define SLAs more accurately.

7.2.1 The QoS MAC layer

Two medium access coordination functions are defined in the original IEEE 802.11 MAC: a mandatory Distributed Coordination Function (DCF) and an optional Point Coordination Function (PCF) which neither support packet priority nor admission control; limitations that motivated researchers to enhance the performance of the WLAN MAC layer and resulted in the release of the IEEE 802.11e standard in 2005 [4]. The IEEE 802.11e standard prescribes how the DCF and PCF functions should be combined for QoS data transmissions. The WiFi alliance certifies a subset of the IEEE 802.11e standard, namely the Enhanced Distributed Channel Access (EDCA) which improves the legacy DCF, and named this Wireless Multi Media (WMM). The QoS guarantees reported in WLAN products today mainly stem from implementing WMM and augmented with a connection admission control scheme; a necessary part not described in the IEEE 802.11e standard.

EDCA introduces four different First-In First-Out (FIFO) queues, called Access Categories (ACs). Different kinds of applications (e.g., background traffic, best-effort traffic, video traffic, and voice traffic) can be directed into different ACs. Each AC behaves as a single DCF contending entity. The stations for which admission control is required transmit a QoS request to the AP containing a traffic specification (TSPEC) of its network resources (e.g., mean/peak data rate, burst size). If the AP accepts the request, it calculates the amount of time for admitted traffic to access the medium, called medium time.

The algorithms used by the AP to make an admission decision and to calculate the medium time are not standardized and designing efficient solutions allow vendors to differentiate their products. The AP sends back to the station a response frame containing a transmission opportunity (TXOP) that indicates how long the station can use the medium for the associated AC and from which the medium time is derived. If a station needs more channel access time for an AC, it has to send a new request to the AP. In contrast to DCF, the contention parameters in EDCA ($Cw_{min}[AC]$, $Cw_{max}[AC]$, $AIFSN[AC]$ and $TXOPLimit[AC]$) are configurable and computed from periodic beacon frames sent by the AP.

EDCA introduces for each AC the Arbitration InterFrame Space (AIFS), replacing the DIFS in legacy IEEE 802.11 DCF systems. Each AIFS is an interval that dictates the time that the medium should be idle before the station can send from the AC and can be computed as follows: $AIFS[AC] =$

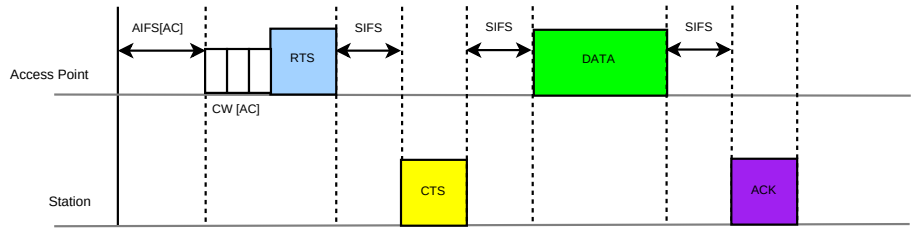


Figure 7.1: Transmission cycle of an IEEE 802.11e-compliant station.

$SIFS + AIFSN[AC] \cdot \tau$, where $AIFSN[AC] \geq 1$ and τ represents the slot time (see Table 2.1). Since $AIFSN[AC] = 2$ corresponds to the parameterization of legacy IEEE 802.11b systems, because $AIFS[AC] = DIFS$, it is clear that high priority traffic in EDCA has precedence over traffic sent by legacy stations. Upon arrival of a new frame the AC draws its random backoff slots from a uniform distribution $[0, \cdots, Cw[AC]])$. At the first transmission attempt, $Cw[AC]$ is set equal to a minimum value $Cw_{min}[AC]$ that can be computed from values sent in the beacon frames. In the case of a failure event, the value of $Cw[AC]$ shall be set to the following value: $2(Cw[AC] + 1) - 1$.

The purpose of using different contention parameters for different queues is to give a low-priority class a longer waiting time than a high-priority class, so the high-priority class is likely to access the medium earlier than the low-priority class. The backoff times of all ACs are decremented each slot time the medium is sensed idle after the $AIFS[AC]$ has expired; if the backoff timer for an AC reaches zero the station may transmit the first frame from that AC. Optionally, the station may first send a small packet (named Request-To-Send, or RTS). If acknowledged by a Clear-To-Send (CTS), the frame can be send. In Figure 7.1 the transmission cycle is shown in which two IEEE 802.11e-compliant stations exchange one MAC data frame.

Since the backoff times of different ACs in one station are randomly generated and may reach zero simultaneously, an internal collision may arise. If this occurs, a virtual scheduler inside every station allows only the highest-priority AC to transmit its frame. Note that QoS requests are made for specific applications, whereas the packets from different applications may arrive at the same AC for transmission. The packets from the AC are transmitted, however, without considering the individual TSPEC values for each application. Consequently, applications using the same AC may harm each other. The transmission opportunities, however, only apply to a single AC.

7.2.2 WLAN overhead expressions to justify new traffic profiles

Overhead is common to make a communication protocol operate properly. The MAC protocol increases the WLAN reliability by including overhead required

for the physical medium and by forcing packet acknowledgments. In practice, the signal strength between each WLAN station and its AP varies frequently as a result of signal propagation effects and decreases when the distance increases. If the signal quality between station and AP declines, the transmission rate of a station may decrease as a result of rate adaptation and consequently the WLAN overhead increases. This increase in overhead soon exceeds a level where the WLAN airtime spent on overhead exceeds the airtime required to transmit or receive the desired data. The IEEE working group addresses the inefficiency by introducing e.g., block acknowledgments and extending the number of frames that can be sent concurrently. These efforts may certainly reduce, but not eliminate the situation.

Bytes per packet	Packets per second	Application rate (bits/s)	IP overhead (%)	Ethernet overhead (%)	WLAN @ 54 Mbit/s overhead (%)	WLAN @ 6 Mbit/s overhead (%)
1250	10	100000	3	5	61	916
625	20	100000	6	11	122	1034
250	50	100000	16	26	305	1385
125	100	100000	32	53	602	1963
25	500	100000	160	264	2978	6650

Table 7.1: Rates and overhead at the application and resulting additional overhead introduced at the IP, the Ethernet and WLAN (IEEE 802.11g) layers (percentages of overhead, experienced by a WLAN station capable of sending at 54 Mbit/s or 6 Mbit/s).

Table 7.1 shows how fixed rates of 12.5 KByte/s at the application layer correspond to the overhead introduced at various protocol layers. The values shown for the IP-layer include the higher-layer protocol overhead for using the Real Time Protocol (RTP), UDP and IP to transport application information. When these packets are transported over Ethernet or WLAN (minimum values, as blocking and retransmissions are neglected), the overhead increases to the values indicated. The values in Table 7.1 are calculated as follows. For IP and Ethernet, the relative overhead is determined by dividing the total protocol overhead (40 bytes for IP and $40 + 26 = 66$ bytes for Ethernet) by the sum of the number of bytes per packet and the protocol overhead.

For the relative WLAN overhead, consider an IEEE 802.11g AP where an associated IEEE 802.11g station uses a Voice over IP (VoIP) application application that creates IP packets containing 10ms voice, using the ITU-T G.711 codec. Packets of 80 bytes are transmitted at a frequency of 100 packets/s in the upstream direction. The packet size offered to the WLAN layer then equals 128 bytes, resulting in a bit rate of 102.4 Kbit/s ($8 \cdot$ (80 bytes voice samples +40 bytes for IP/RTP/UDP +8 bytes for LLC/SNAP)$= 8 \cdot 128 \cdot 100$ packets per second = 102.4 Kbit/s).

The overhead that is introduced up to the WLAN-MAC layer can be calculated by using the parameters from Table 7.2 in (2.1) and (2.2) to subsequently obtain

Parameter	802.11a	802.11b	802.11g	802.11n	802.11g mixed-mode
Cw_{min}(slots)	15	31	15	15	15
MAC(bits)	246	224	246	246	246
τ	9μs	20μs	9μs	9μs	20μs
SIFS	16μs	10μs	10μs	16μs	10μs
DIFS	34μs	50μs	28μs	34μs	50μs
EIFS	90μs	364μs	342μs	90μs	364μs
PHY	20μs	192μs	26μs	20μs	26μs
ack(bits)	134	112	134	134	134
T_p	1μs	1μs	1μs	1μs	1μs
TS(bps)	$54 \cdot 10^6$	$11 \cdot 10^6$	$54 \cdot 10^6$	$54 \cdot 10^6$	$54 \cdot 10^6$
TS_c, TS_m(bps)	$24 \cdot 10^6$	10^6	$24 \cdot 10^6$	$24 \cdot 10^6$	10^6
STT	4μs	NA	4μs	4μs	4μs
RTS(bits)	182	160	182	182	182
CTS(bits)	134	112	134	134	134

Table 7.2: Summary of important parameters in IEEE 802.11 protocols, including RTS/CTS and mixed 802.11b/g mode operation.

in (2.7) from Section 2.2.3 the time needed to transmit higher-layer protocols in a MAC-acknowledged IEEE 802.11g MAC Protocol Data Unit (MPDU). By coincidence the overhead of the higher-layer protocols X_{hloh} is the same for TCP data segments as for voice packets transmitted over RTP and UDP. In the case of using the optional LLC/SNAP encapsulation as outlined in Section 2.2.4 to transmit voice packets over RTP and UDP X_{hloh} equals 384 bits. The WLAN MAC subsequently adds more overhead to the voice packet, requires additional interframe times to wait for and the transmission of acknowledgment frames, as explained in Section 2.2.2. When assuming that the channel reservation mechanism RTS/CTS is switched off and that the transmission rates of the data and control frames are set to 54 Mbit/s and 24 Mbit/s, respectively. Based on these parameters we obtain for the considered VoIP application an equivalent data rate of almost 0.7 Mbit/s. Note that the 54 Mbit/s is the highest support transmission rate supported by the IEEE 802.11g standard and that 24 Mbit/s is the highest mandatory rate that stations must support [5] and is therefore often the highest rate configured in the BSS basic rate set.

If, however, due to rate adaptation the transmission rate for data and control frames of one particular station degrades to 6 Mbit/s, the resulting equivalent data rate as observed by another station that is able to transmit at 54 Mbit/s would approximately be $1,7$ Mbit/s, an increase of more than 160%. Note that all values are calculated as observed by an associated station that could send at 54 Mbit/s. This also explains why - in contrast to the others - the overhead in the case stations transmit at 6 Mbit/s does not show linear behavior over the

rows.

As can be observed from Table 7.1, the overhead introduced by WLAN is a decisive factor for achieving the guaranteed application data rate at IP layer, and hence an important parameter to consider when the aim is to guarantee the users QoS experience. The remainder of this chapter proposes multi-service traffic profiles and explains how they can be used to compute the WLAN airtime, applying the above calculations more formally.

7.2.3 Service level agreements in WLAN

A SLA defines the basis for an understanding between two parties for the delivery of a service. In the context of this chapter, the service consists of the traffic volumes exchanged between two parties: WLAN station and AP. The SLA defines the conditions of the tolerated traffic volumes within a time interval. Often, the SLA specifies committed and peak rates, and an additional tolerance factor such as e.g., the maximum burst size.

If a station moves away from the AP, the WLAN rate adapts automatically and if that occurs, the station will consume more airtime capacity to maintain the same rate at the IP layer and the layers on top of the IP layer. This may degrade the performance of other users. If a SLA is defined at the WLAN level, the WLAN resource consumption can be kept constant and degradation of other WLAN stations can be prevented. This requires that shaping and policing functions operate at the WLAN level.

Quality of service requires solutions in both horizontal (between network APs) and vertical (between protocol layers) directions are needed to reach the final goal: to give the applications the required network capacity such that users experience the desired and constant quality. It is the users experience (from mouth-to-ear and eye-to-eye) that in the end determines if the desired quality of service level is obtained. Since terminals are commonly equipped with various cards to connect to the network and the IP layer is the first common layer for all applications, the IP layer is the desired layer to negotiate QoS parameters for horizontal and vertical interworking. Moreover, the parameters needed by e.g., various QoS standards to compute the quality of experience for various applications (e.g., for voice ITU-T recommendation G.107, and for video ITU-T recommendation P.59) are measured at the IP layer. Note that the situation that the performance of a station degrades as a result of other stations moving away from the AP is only relevant if the other stations also require the transmit capacity at that time. If not, there is no harm in letting a station consume a relatively large part of the WLAN airtime capacity.

Clearly, if a SLA concerns the traffic rate at the IP layer, the capacity demand on lower protocol layers should be known to make any form of guarantee. The solution found is to define the SLA at the IP layer and subsequently detect that

other stations cannot meet the committed rate assigned to them, taking action to prevent or resolve performance degradation. Regular shaping and policing mechanisms can be used to make sure that the user conforms to the SLA. Mappings to WLAN characteristics given in the previous paragraph are taken into account to compute the WLAN overhead. The underlying assumption in this mapping is that each station can be treated as if it is the only station in the coverage area of an AP. Section 7.4 provides a foundation for this assumption and also shows how the QoS can be maintained while channel variations may occur.

7.2.4 Multi-service traffic profiles and using them to compute airtime

Section 7.2.2 argues that for WLANs the traffic rates in a traffic SLA require more granulation to guarantee and maintain a desired QoS level for all stations. The solution found is to replace each rate (committed, peak, mean) within a traffic SLA by a multi-service traffic profile that consists of two traffic values, namely the mean packet size (BPP, or mean Bytes Per Packet) and mean packet frequency (PPS, mean Packets Per Second), measured at the IP layer. Note that the SLA needs to define a (BPP, PPS) parameter pair for the up-and downstream directions separately. Taking the product of a (BPP, PPS) pair results again in a bit rate.

A first example of a traffic profile includes the sending of WLAN management beacon frames, where the PPS is associated with the beacon interval (often approximately 100 ms, resulting in $PPS = 10$) and BPP equals the size of the beacon (i.e, 16 bytes $< BPP <$ 2304 bytes). An applications network requirement can be expressed by a traffic profile. For real-time voice and video applications, the traffic profile is natural and straight forward since in general codecs define the constant rate of packet segmentation. A set of n multi-service traffic profiles (BPP_i, PPS_i, for $i = 1, \ldots, n$) can be concatenated to a single traffic profile (BPP^*, PPS^*) to define the users overall QoS profile, as follows:

$$PPS^* = \sum_{i=1}^{n} PPS_i, \tag{7.1}$$

$$BPP^* = \frac{\sum_{i=1}^{n} BPP_i \cdot PPS_i}{PPS^*}.$$

The WLAN transmission cycle can be used to compute the airtime percentage associated with each multi-service traffic profile. This is the same reasoning as used to compute the overhead in Section 7.2.2 and is proposed in e.g., [40] for computing the maximum number of simultaneous voice calls that can be supported by a single WLAN AP under static WLAN conditions. Let $T_k(x)$ denote the time needed to successfully transmit a packet of x bits application data, given k retransmissions were needed ($k = 0, \ldots, N$). The parameter N reflects the maximum number of retransmissions; if more than N attempts are

needed to transmit a packet, it is discarded.

Applying the WLAN transmission cycle for a WLAN client i that sends and receives with transmission rate TS_i and to whom the multi-service traffic profile (BPP_i, PPS_i) was assigned in, say, the downstream direction, the following expression for $T_k(x)$ yields (assuming that the maximum contention window size is reached for $k = N$):

$$T_k(x) = T_d(x) + T_c + 2^{k-1} C w_{min} \tau + \delta(RTS) + RTSA, \tag{7.2}$$

where $T_d(x)$ and T_c are defined in Chapter 2 by (2.1) and (2.2) respectively. The applicable MAC parameters are specified in Table 7.2. For sake of simplicity, the propagation delays of the packet transmissions can be omitted due to their very small influence. The function $\delta(RTS)$ denotes the Dirac function and is zero if RTS/CTS channel reservation is disabled and equals one otherwise. In the case the RTS/CTS channel reservations are enabled (or IEEE 802.11g stations insert CTS-to-self frames in the presence of IEEE 802.11b stations in the BSS), additional overhead associated with the RTS (160 bits) and CTS (112 bits) messages (expressed by the parameter $RTSA$) should be added to determine $T_k(x)$.

The use of RTS/CTS and CTS-to-self are protection mechanisms against interference, of which CTS-to-self shall be used by IEEE 802.11g/n-compliant stations in the presence of legacy IEEE 802.11b stations that are associated with the same AP. Both protection mechanisms may have a large influence on the medium efficiency because the frames are sent at low transmission rates and additional interframe spacing is involved. When there are no legacy stations associated with an AP and RTS/CTS channel reservations are used, $RTSA$ may be defined for IEEE 802.11a/g/n-compliant stations as follows:

$$RTSA = 2\,(PHY + SIFS) + \left\lceil \frac{RTS}{TS_c \cdot STT} \right\rceil \cdot STT + \left\lceil \frac{CTS}{TS_c \cdot STT} \right\rceil \cdot STT, \tag{7.3}$$

with TS_c defined by:

$$TS_c = min\,(TS_i, max\,(BSSBasicRateSet))\,, \tag{7.4}$$

where TS_i corresponds to the current transmission rate of the station and the BSSBasicRateSet represents the set of rates configured at the AP.
When one or more legacy IEEE 802.11b stations have associated with an AP, IEEE 802.11g/n stations must operate in backwards compatibility mode and apply the CTS-to-self protection mechanism, which leads to the following $RTSA$ for the associated non-legacy stations:

$$RTSA = PHY + SIFS + \left\lceil \frac{CTS}{TS_c \cdot STT} \right\rceil \cdot STT, \tag{7.5}$$

where TS_c is defined in (7.4).
In the case of using RTS/CTS channel reservations in a network where both legacy and non-legacy stations are associated, the following $RTSA$ applies:

$$RTSA = 2\,(PHY + SIFS) + \frac{RTS + CTS}{TS_c}, \tag{7.6}$$

where TS_c is used from the BSSBasicRateSet corresponding to IEEE 802.11b legacy systems:

$$TS_c = min\,(TS_i, max\,(11bBSSBasicRateSet))\,, \tag{7.7}$$

The exponentially increasing contention window at each retransmission forms the basis for our assumption that collisions at the wireless medium occur independently. If p_i denotes the probability that a packet from the profile (BPP_i,PPS_i) collides, the probability that k ($k = 0, 1, \ldots, N$) retransmissions are needed becomes a truncated geometric distribution. Since $T_k(x)$ is defined under the condition that k retransmissions were needed, the percentage of airtime κ_i associated with this traffic profile is obtained as follows:

$$\kappa_i = \frac{PPS_i \sum_{k=0}^{N} T_k(x) p_i^k}{\sum_{k=0}^{N} p_i^k}. \tag{7.8}$$

Note that the parameter $T_k(x)$ includes the contention period a WLAN station is forced to wait before it can transmit a frame. The empty slots are necessary to decrease counters at the WLAN stations, but from the perspective of the wireless medium they do not consume any resources. To quantify this aspect, the part that relates to the contention window needs to be adjusted, where the load on the medium is a perfect candidate for this adjustment. To keep the formula general and to e.g., allow to re-use it to compute net airtime consumption, an adjustment factor α ($0 < \alpha < 1$) is introduced, resulting in the following adjusted formula for $T_k(x)$:

$$T_k(x) = T_d(x) + T_c + \alpha C w_{min} \tau 2^{k-1} + \delta(RTS) + RTSA. \tag{7.9}$$

Substituting the expression for $T_k(x)$ from (7.2) into (7.8) and making use of the fact that for any $a \neq 1$ and finite n, $\sum_{i=0}^{n} a^i = (1 - a^{n+1})/(1 - a)$, the following expression for κ_i arises (p_i in $[0, 1>$; $p_i \neq 1/2$):

$$\begin{aligned} \kappa_i &= PPS_i\,(T_d(x) + T_c + \delta(RTS) + RTSA) \\ &\quad + \frac{PPS_i \alpha \tau C w_{min}\,(1 - p_i)\left(1 - (2p_i)^{N+1}\right)}{2\left(1 - p_i^{N+1}\right)(1 - 2p_i)}. \end{aligned} \tag{7.10}$$

For the exceptional situation $p_i = 1/2$, the following expression for κ_i holds:

$$\begin{aligned} \kappa_i &= PPS_i\,(T_d(x) + T_c + \delta(RTS) + RTSA) \\ &\quad + \frac{PPS_i \alpha \tau C w_{min}\,(1 - p_i)\,(N + 1)}{2\left(1 - p_i^{N+1}\right)}. \end{aligned} \tag{7.11}$$

	Page body (bytes)	images per page (images)	AV. Img size (bytes)	Total Img (bytes)
http://www.swisscom.com/	61998	18	6854	123370
http://www.francetelecom.com/	55429	51	1761	89799
http://www.npt.no/	57882	18	2891	52030
http://www.ismb.it/	40492	29	2061	59762
http://www.tu-berlin.de/	21765	23	6341	145837
http://www.motorola.com/	23576	20	1827	36532
http://www.bell-labs.com/	24538	29	4481	129938
http://www.tid.es/	13375	22	4335	95375
http://www.eurasip.org/Proceedings/Ext/IST05/	16457	25	5331	133278
Average	35057	26	3987	96213
Minimum	13375	18	1761	36532
Maximum	61998	51	6854	145837

Table 7.3: Webpage measurements.

The remaining parameters needed to compute κ_i are α and p_i, i.e., the adjustment factor and the probability that packets of the multi-service profile (BPP_i,PPS_i) collide. Section 7.3 provides insight in how this parameter can be determined.

7.2.5 Traffic profile examples and elastic traffic

For real-time applications the traffic profile definition is natural as these applications often result in a constant flow of data packets propagating through the network. Elastic data applications can adapt to time-varying available network capacity via a feedback control mechanism, e.g., part of the transmission control protocol. Typical elastic data applications are file transfers, e-mail web browsing applications. This section shows how the traffic profiles can be used to define elastic traffic. A characterization of a browsing application was made by using the statistics of realistic set of Internet pages shown in Table 7.3. Having determined a representative page size, the effect needs to be determined on the generated network traffic as a result of a WLAN user browsing these pages. The key parameter in browsing applications is the page response time. Figure 7.2 shows the simulated network configuration where one user (Usr) is connected through a (IEEE 802.11b) WLAN AP. The web server (Server) is located behind a set of routers and an IP network which together add a uniformly distributed delay between 100 and 200ms to all packets.

For a constant page inter-arrival time of 100 seconds between successive page requests the simulation outcomes stated in Table 7.4 are obtained. The simulation outcomes are the traffic statistics obtained just above the WLAN layer (and include the IP payload and header) and showed an average page response time of 8.69 seconds. Figure 7.3 illustrates the web browsing cycle between two consecutive page requests; during period T_1 the page is loaded and traffic is

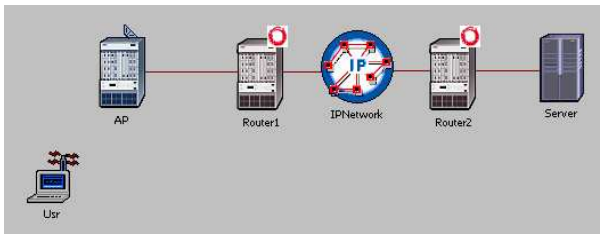


Figure 7.2: Simulated network infrastructure.

Parameter	Upstream	Downstream
Average packet rate (PPS)	0.77	1.14
Average packet size (BPP)	174.84	1396.81

Table 7.4: Traffic profile of browsing session, measured at the IP-layer and obtained through simulation (packet rate and data rate statistics are obtained with 99% confidence intervals $< 0.2\%$).

sent in the upstream and downstream directions to realize the page download. During period T_2, the medium is idle. The influence of this silent period needs to be taken into account to compute the packet frequency during the peak period. By scaling the average packet rate measured in the simulation network with the inverse duty cycle of the browsing session $((T2 + T1)/T1)$, the traffic characteristics can be obtained for the period T_1.

Any other period length L with $(T_1 \leq L \leq T_1 + T_2)$ can be selected to change the expected users page response time and associated burst size, resulting in a scaling factor of $(T_1 + T_2)/L$. With the given page request inter-arrival time of 100 seconds and the average page response time of 8.69 seconds, the average packet rate during period $L = T_1$ is obtained by scaling the average packet rate with approximately 11.5 times to yield in downstream 13 PPS and in upstream direction 9 PPS.

Based on these results, a set of example profiles for the various performance levels of the browsing applications is obtained, shown in Table 7.5.

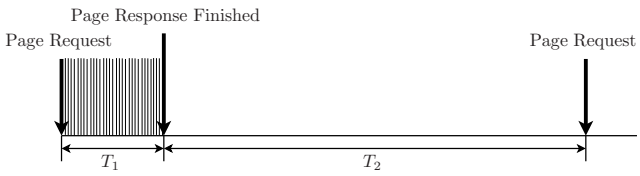


Figure 7.3: Web browsing cycle.

Application	Upstream		Downstream	
	BPP	PPS	BPP	PPS
High-quality voice	1024	100	1024	100
Medium-quality voice	464	100	464	100
High-quality browsing	175	9	1397	13
Medium-quality browsing	175	4	1397	7

Table 7.5: Example multi-service traffic profiles for voice and browsing applications. High-quality voice is defined by G.711 (64 Kbit/s) voice samples with a length of 10ms. Medium-quality voice is defined by voice 10ms packets encoded with G.729 codec (8 Kbit/s). For high-quality browsing $L = T_1$; for medium browsing $L \approx 2 \cdot T_1$ was selected.

The simulated scenario does not provide an in-depth study of all possible situations that can occur and their associated parameters. Increasing the network load by allowing more users that compete for accessing the medium increases the page response time and hence the PPS. The BPP in the upstream and downstream directions will not change. For UDP applications such as real-time voice conversations, the profile information only changes in the case codecs are used that change real-time. In most cases, the codec rate is static (e.g., DVD; MPEG2) and hence known at the start of the real-time application, which can be reflected in the PPS of the profile.

7.3 Guaranteeing QoS for WLAN networks

Providing guarantees for the WLAN channel environment in a commercial setting is challenging since it operates in unlicensed spectrum where interference from other devices is common, and the channel propagation properties can vary widely with movement of the wireless devices or objects in their vicinity. This variability makes it exceptionally difficult to maintain the quality of experience but has to be accounted for to offer and guarantee QoS. The AP needs to adjust the scheduling and airtime allocations to compensate, possibly impacting the QoS delivered to one station when the conditions of the connection to another station with a higher-priority stream become degraded.

The IEEE 802.11e standard describes that the AP should grant QoS with specific QoS parameters appropriate to current channel conditions, and meets the request of a particular traffic stream or class to the best extent possible but does not describe how this should be achieved. This chapter explains how this can be realized by granting each station a QoS budget that corresponds to a user profile, specified as a pair (BPP, PSS). Two user profiles are distinguished: peak and committed. The peak profile indicates the level that should not be exceeded, whereas the committed profile indicates what clients may expect from the network.

7.3.1 Computing and assigning traffic budgets to realize QoS

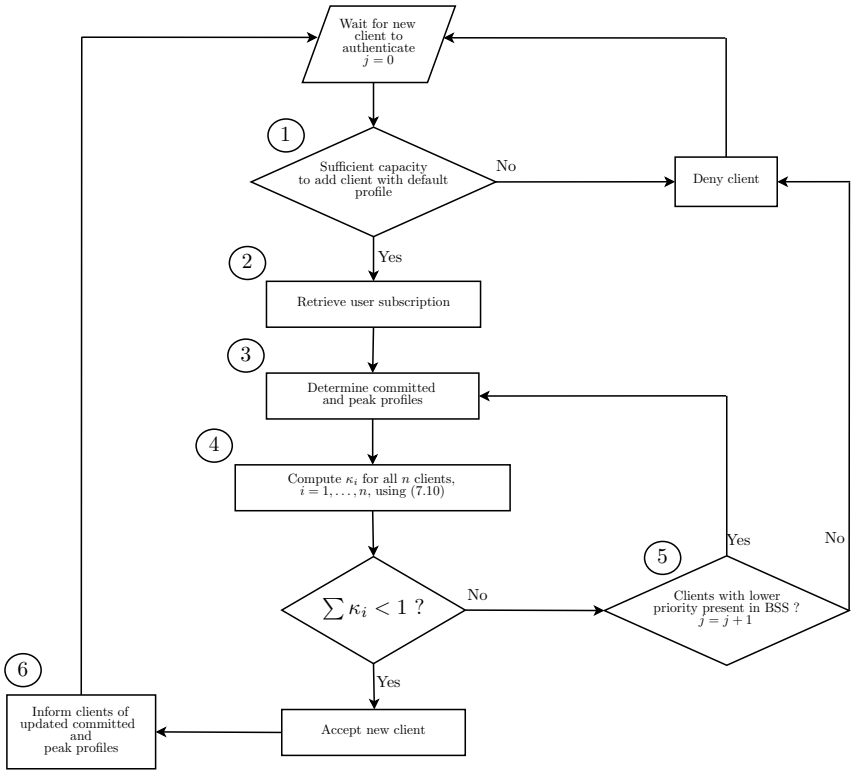


Figure 7.4: Adding a newly arriving WLAN client and taking into consideration user continuation.

We explain how the user profiles are determined and assigned by considering the case of a newly arriving WLAN client. The solution runs on a WLAN AP and the process is depicted in Figure 7.4. The numbers used in this figure correspond to the explanation below.

1. The QoS solution receives the notification of a new arrival from a security proxy. It determines whether there is sufficient capacity to add the newly arriving client, using a default profile which can be stored locally. If there is not sufficient capacity, the proxy can stop the authentication process and can report to the client that authentication failed. The decision is the same as explained in steps 4 and on-wards.

2. A users subscription consists of a set of traffic profiles, stored in e.g., the Home Location Register (HLR) or an AAA server and can be obtained as part of e.g., the authentication process making use of protocols such as Radius or Diameter (see also Section 7.4.1).

3. A network node aware of the WLAN APs (e.g., the residential gateway in a users home or the AP controller of a WLAN campus network or a primary node in a WLAN wireless mesh) ranks the subscriptions in descending order according to the airtime needed and computes the station's QoS budget by taking into account the subscription of the newly authenticated client as well as the traffic budgets of the clients already assigned to the same AP and which are maintained while the users are associated. Various rules may influence the decision to label a profile out of the users subscription as committed or peak, possibly depending on the users priority level, the medium type of the AP and local policies. After ranking, the first $+j$ and the second $+j$ profiles of the subscription can be assigned as peak and committed profiles, respectively. The parameter j is increased every time step 5 is passed.

4. To compute the airtime capacity κ_i for all clients, the set of multi-service profile parameters (BPP, PPS) are determined in the previous step. The WLAN transmission rate can be obtained from the AP, by e.g., SNMP. The remaining parameters to compute κ_i are the blocking probability p_i and the factor α. For determining p_i, we observe that one device (namely the AP) sends all downstream traffic and hence experiences in most situations the highest blocking probability. The airtime requirements of both all upstream and all downstream profiles need to be fulfilled to deliver the desired quality of experience to each user. It is therefore recommended to set p_i for all clients equal to the blocking probability seen by the AP. By using committed profiles to compute (PPS^*, BPP^*) in accordance with (7.1) and consequently feeding the result in (7.10) with $\alpha = 0$, dividing the resulting κ by 2 provides an accurate estimate for the blocking probability. The factor 2 can be explained by the fact that we consider only the traffic from the AP. Note that the sending of beacon frames should be included in the computation of (PPS^*, BPP^*). For the parameter α, the net load of the medium can be taken, i.e., $\alpha = \kappa$, computed by (7.10) with $\alpha = 0$ (after all, the net load of the medium does not need to include the back off period).

5. If the sum of the airtime capacity consumed by each individual client exceeds unity, there is not sufficient airtime to add the newly arrived client.

However, the QoS policy may dictate that clients with less priority should make room for the newly arrived client with higher priority. To realize this, the set of clients with lower priorities should be determined and their peak and committed profiles can be re-determined. This can be realized by selecting the j^{th} element of the subscription, where the parameter j is increased for clients with lower priority. This can be repeated until either the sum of all $\kappa_i > 1$. If this stage cannot be reached, the parameter j for the newly added user can be increased, granting the user access with a lower service profile than desired.

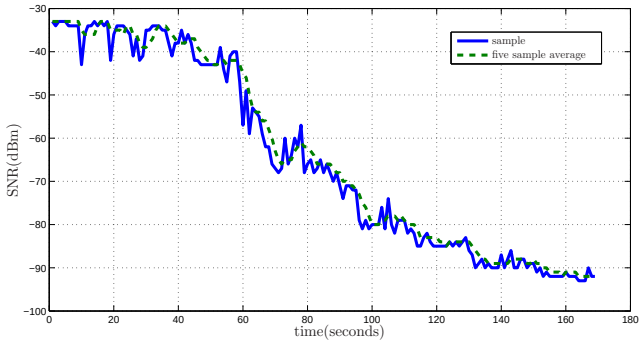
6. The newly added client need to be informed about the granted QoS budget (i.e., assigned multi-service profile (BPP, PPS). In addition, the allocated profiles of other clients with lower priority may have changed as a result of passing step 5 and hence require updates. The fields of the IEEE 802.11e protocol can be used to inform the clients. For clients that are not compliant with the IEEE 802.11e protocol, only the policing parameters may need to be updated. These users may see the effect by degradation of the quality of service without having received a proper notification.

7.3.2 Maintaining QoS

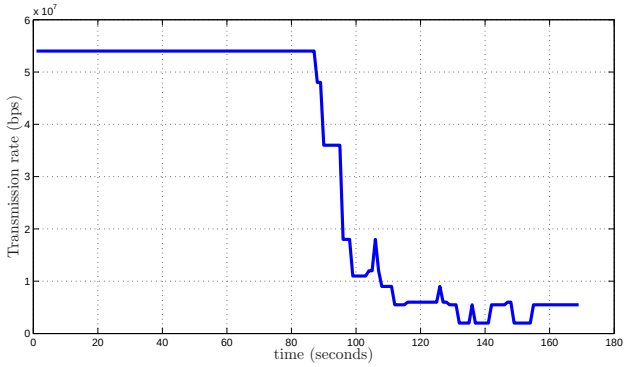
The capacity offered by WLAN channels may fluctuate as a result of transmission rate adaptations. Consequently, the airtime assigned to users may turn out to be insufficient to provide the desired level of QoS to the user. To prevent this, the AP (or a module communicating with the AP via e.g., SNMP) can monitor the transmission rate and the bytes sent to and received from each station and use this as multi-service profile (BPP, PPS) to compute the total airtime consumption. The same formula (7.10) can be used to compute these values, meaning that the proposed solution operates on both a relatively short time scale (varying channel conditions occur on ms) as well as a relatively long time scale (user admission control functions operate on a minutes time scale). The same process as depicted in Figure 7.4 applies, but this time the module that monitors the WLAN channel alarms that the airtime needs to be redistributed.

Since transmission rates may alter, multi-service traffic profiles need to be recomputed and re-assigned to maintain the desired QoS for high priority clients. The channel conditions causing these rate adaptations may vary quickly and consequent actions of the QoS solution that are based on measurements from the past may deliver the desired effect in time. Figure 7.5 illustrates the behavior of a IEEE 802.11g device moving at walking distance away from the AP in an indoor office environment; the transmission rate adaptations may occur unexpectedly and quickly succeed each other. As the wireless LAN standard has not defined the conditions for performing rate adaptation it is device-specific and may vary widely between various vendors and types. To increase the responsiveness of the proposed QoS solution, a low-complexity transmission rate prediction algorithm was developed [48] to anticipate future transmission rate

adaptations of all associated stations. The basis of the prediction algorithm is determining the expected average Signal-to Noise Ratio (SNR) for the next time interval and mapping that value to the expected transmission rate. A transmission rate matrix M is used for this goal. Element (r, k) in matrix M contains the average SNR value for which adaptations have occurred from transmission rate r to transmission rate k. These values are periodically retrieved on the AP for all associated stations as part of the SNMP measurements. The expected transmission rate for the next measurement interval of a concerned station is obtained with a two-step approach: first, an expected average SNR value of the next interval is determined by linear extrapolation the last two average SNR measurements, and secondly, the expected transmission rate is determined by mapping the expected average SNR with the recorded transmission rate adaptations. The resulting per-station airtime consumption is subsequently determined by using the predicted transmission speed in (7.2).



(a) Signal To Noise (SNR) values.



(b) Transmission rate values.

Figure 7.5: Measured (one sample per second) and averaged samples of SNR values (as observed by the AP) and the corresponding transmission rate of an IEEE 802.11g device moving at walking distance away from the AP in an indoor office environment; the transmission rate adaptations may occur unexpectedly and quickly succeed one another.

Predictions are made by the prediction algorithm while the transmission rate matrix is recorded. As the number of transmission rate adaptations increases, the prediction for the tagged station becomes more accurate. Rather than starting from an empty matrix, initial predictions may be based on a generalized matrix. If the QoS solution detects that the total airtime consumption exceeds a predefined threshold, the AP concludes that it is not able anymore to commit to the assigned profiles. The preferred actions to restore the QoS level of the station(s) in danger is to reduce the peak profiles of either all clients, or of only those that exceed their committed profiles (e.g., set to the committed profiles), as dictated by the QoS policy. Note that reducing the profiles also requires updating the policing mechanisms. To prevent oscillation, the values should not

be updated too frequently, say only once in three seconds.

7.3.3 Numerical validation

If the WLAN channel conditions are static, our proposed solution is comparable with an admission control schema. In those circumstances, Medepali et al. show in [85] that the resources consumed by a voice call over WLAN can be computed accurately by assuming the WLAN consists only of the station of interest. They compute the resource consumption associated with voice over WLAN call legs by taking the weighted product of the number of packets generated in each state of a 4-state Markov chain including idle and busy times (the ITU-T P.59 conversational speech model). For the generic notion of the concatenation of profiles, the conversational speech model cannot be used as a traffic profile neither contains idle nor busy times. The resource consumption for voice applications can be obtained by expressing the codec details of these applications in multi-service traffic profiles (BPP,PPS). Table 7.6 compares the results obtained by using (7.9) with the simulation results reported in [85] for voice assuming highest transmission rates, revealing an accurate match.

Codec interval(ms)	PPS	BPP	802.11a or 802.11g		802.11b	
			Simulation	Analysis	Simulation	Analysis
10	100	80	55	57	11	12
20	50	160	105	105	21	22
30	33.3	240	149	148	30	31
40	25	320	185	187	38	39
50	20	400	220	221	44	46

Table 7.6: Comparing analysis resource consumption with simulation results for number of voice call lags on WLAN found in [85].

7.4 Lessons learned from implementing multi-service traffic profiles

7.4.1 Use case

The transmit capacity of the access link that connects homes to the Internet is often not fully used, either because users select a low subscription or because the user is not constantly on-line, leaving a surplus capacity. Reference [94] explains that the WLAN equipped home network facilities may become an interesting asset for fixed-wireless convergence and have the potential to realize a large international WLAN network, solving the scattered subscriptions with minimum investments. The multi-user profiles and the solution to compute, assign and maintain a QoS budget were applied for this use case. Reference [95] explains how mobility and security (see also [62]) can be realized effectively.

Figure 7.6 depicts the networks and players required for public users to gain

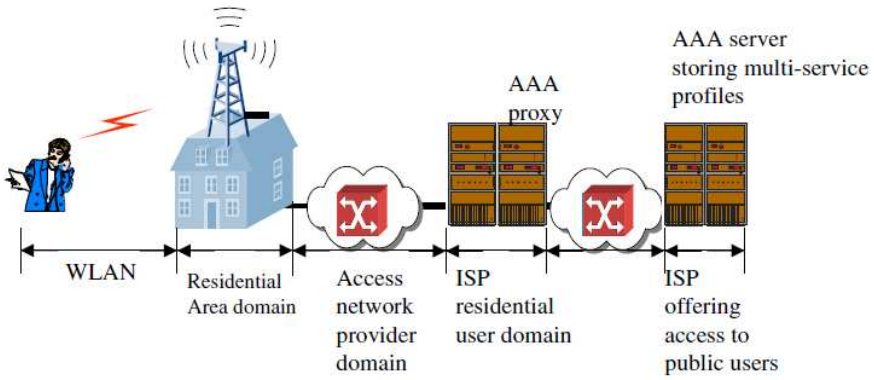


Figure 7.6: Use-case: offering access to casually passing users via residential areas.

access via their WLAN terminal. The residential area consists of one or multiple WLAN APs, a router and a modem, together referred to as the residential gateway. For implementing the use case a Cisco AP 1131AG was used, supporting multiple SSIDs and hence allowing a natural way to separate the public traffic from residential traffic on the WLAN.

The multi-service traffic profiles were used to specify home and public user subscriptions and also to compute the network load of the WLAN medium, based on packets sent/received information obtained from the AP. The local AAA proxy in the residential gateway stores subscription profiles for the residential users whereas the AAA server of the ISP stores the subscription profiles of the public users. The algorithms computing the QoS budget were implemented by a module referred to as the QoS keeper, located in the residential gateway and fed by the subscription profiles received from the AAA proxy in the RGW. In the case of public users, these profiles were received as part of the Radius vendor specific attributes of the public users final and successful authentication message.

The QoS budget available for public users was determined by reducing the total WLAN capacity with the capacity committed to residential users, taking into account the possible limitations of the capacity offered by the broadband access connection. A 100 Mbit/s Ethernet back-end connection was used, eliminating bottlenecks at the side of the fixed infrastructure. The multi-service profiles were composed of a mix of voice and browsing applications in various quality levels as well as profiles with a multitude of voice applications, representing users with high resource requirements. WLAN specific input parameters such as the type of network used, the amount of traffic sent/received, and the transmission rate of individual stations were obtained by communicating with the AP via

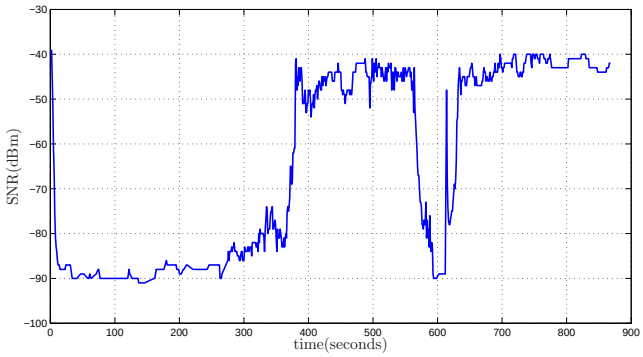
SNMP.

7.4.2 Effectiveness of the solution

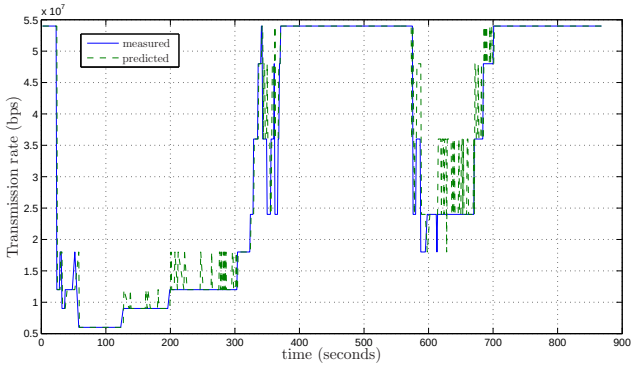
The medium was monitored, resolving transmission rates as well as bytes per packet and packets per second of each individual user by regularly polling the AP. Based on this information, part of the profile selection process can be applied to obtain the airtime consumption (or load) induced by each station and, hence, of the ensemble of associated stations. If the load exceeds a predefined maximum, consequent actions are triggered to re-distribute the wireless medium resources appropriately over the stations to avoid violating QoS guarantees. One of the issues in applying this method is that the transmission rate is not always a representative value of those used during the polling period, which could lead to deviations in determining the load for the measured QoS profile (which represents all traffic exchanged). The effect of this was most profound when high traffic volumes were sent/received in combination with sudden drops in the transmission rate. This issue was circumvented, firstly, by taking the smallest polling interval (500 ms) that the SNMP interface would reasonably allow and, secondly, by applying a sliding average over several of the most recently obtained load values. As the averaging could slow down the system response, the third measure taken was to apply a method for transmission rate prediction to anticipate on future transmission rate declines/increases and to incorporate the effect in the sliding window load average. These measures also counteracted the effects when stations would quickly drop their rate in motion and restore their rate when becoming stationary again or when oscillating between rates in stationary condition. However, the preferred solution to achieve a quick and accurate system response is to have the channel monitoring functionality integrated with the AP, or to use an interface that allows faster data transfer.

7.4.3 Learning and prediction

In a simple experiment, an IEEE 802.11a device moves two times straight from- and towards the AP at walking speed in an indoor office environment. In Figure 7.7 it can be observed from the measured SNR and the measured transmission rate that the device moves away from the AP at approximately 0 seconds and returns at 370 seconds, it again moves away at 570 seconds and is in close range at 700 seconds. During the experiment the prediction algorithm learns the transmission rate adaptations and makes a prediction of the transmission rate for the next measurement interval from the eight available values. The predicted transmission rate was correct for 86.9% of the measurements made.



(a) Signal To Noise (SNR) values.



(b) Transmission rate values.

Figure 7.7: Measured SNR values (as observed by the AP) and the corresponding transmission rate values and the predictions thereof for an IEEE 802.11a device moving at walking distance away from and to an AP in an indoor office environment.

Practical issues arose due to the power-down behavior of some network interfaces; some perform a sudden power-up and transmission rate adaptation to a lower rate (e.g. 11 Mbit/s while normally operating at 54 Mbit/s). These transitions ruin the transmission rate adaptation matrix and hence the predictions correctness, for which simple practical solutions exist.

7.4.4 QoS policies

As soon as the consumed WLAN load would exceed a predefined maximum system load and the wireless network would need to be re-distributed among the wireless users various considerations could be taken into account, for example, the currently assigned QoS profile or the number of subsequent (re)distribution

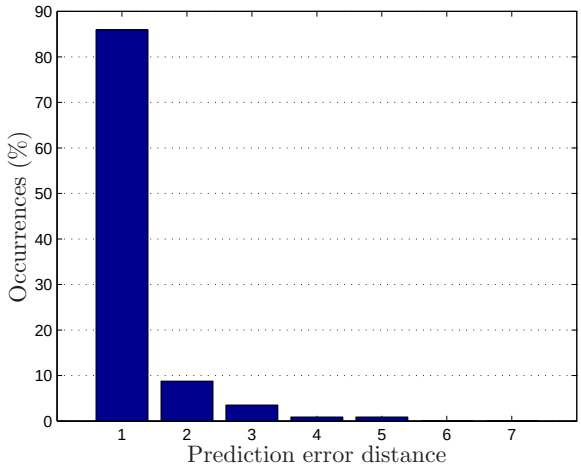


Figure 7.8: Distance between the actual and predicted transmission rate in the case of an erroneous prediction for the walking experiment.

attempts to lower the system load below threshold. When including several degradation levels within the user profiles, graceful restoration of the system load can be combined with a gradual reduction of those QoS guarantees that would require correction anyway. As different preferences or business models to operate such wireless LAN may be desired, two categories for QoS policies are implemented:

1. User class prioritization: for residential and public user airtime allocation. One can pre-allocate a fixed amount of airtime specifically for residential users or to assign precedence residential users over others.

2. Intra-user-class prioritization: to define the method for allocating resources within a user class. In this way, users can have precedence over others in the same class. By basing the priority class on the number of traffic profiles included in each user subscription a natural set of priority users can be defined without the need to predefine priority classes (gold/silver/bronze). Alternatively, as the transmission rate is one of the parameters to compute traffic budgets, priority can be based on vicinity to the AP or on equal sharing.

7.5 Conclusions and further research

This chapter shows that the overhead introduced by the WLAN protocol and physical layer causes that the parameter bits/s does not provide accurate insight in load conditions and/or traffic demands in WLANs. A multi-service

traffic profile is introduced to provide this insight. Although the solution is made suitable for WLAN networks, the approach of using multi-service traffic profiles to realize and maintain QoS guarantees is also applicable to other rate-adapting networks. The traffic profiles are used to introduce the notion of a QoS budget, allowing terminals to determine locally whether current network consumption and/or a newly started network greedy application fit within the traffic contract. The QoS budget consists of two levels, namely committed and peak, which can in turn be associated with the IEEE 802.11e traffic classes. Based on the time sensitivity of each application, clients can store packets in separate queues which are served with different priorities, ensuring prioritization of time sensitive connections. The QoS budget forms a traffic contract for the IP layer and dictates the size and the frequency of packets offered to the WLAN layer.

The incentive that terminals shape outgoing traffic according to the assigned QoS budget can be obtained naturally by applying policing mechanisms, dropping packets that exceed the assigned QoS budget. By specifying the QoS budget on IP level, the standard shaping and policing mechanisms can be used to prevent violating the budget and to drop and/or mark packets in the case stations do not respect the QoS budget. Charging the various priority levels differently can be used to de-motivate stations to mark all packets to the wrong priority level. Fading effects as well as the ability of a WLAN station to move to and away from an AP raise the situation that WLAN stations experience varying network resources. In addition, allowing too many users at the same AP may cause insufficient WLAN airtime to satisfy the users quality of experience. The algorithm proposed to maintain and guarantee the QoS levels at the IP level such that the users quality of experience can be fulfilled. It realizes this by taking into account the varying WLAN channel conditions and shows accuracy within the 5% range, when compared to computing the maximum voice channels that can be sustained concurrently. The same algorithm is used to compute the per-client WLAN airtime consumption, from which the overall WLAN load is computed. The complexity of the algorithm is linear in the stations associated within the BSS. The QoS level of stations can be kept within desired bounds by monitoring the WLAN medium and redistributing the per station airtime budget when high consumption dictates, realizing predictable QoS levels for high priority WLAN users. The WLAN stations can be informed by the assigned QoS budget as part of the authentication process. If QoS budgets can be re-assigned, each station is informed about their network budget and hence serves as an alternative for the per-flow connection admission control which is challenging to meet for each flow individually. The IEEE 802.11e standard is identified as a candidate to inform stations about changed QoS budgets, providing a total QoS budget per WLAN station and not a per-flow guarantee. This allows user-differentiation on their resource consumption; QoS policies can dictate how the available airtime should be distributed over all associated terminals.

An interesting topic for further research is to apply the proposed solution for traffic control to other network technologies that utilize transmission rate adaptation. The proposed transmission rate prediction algorithm demonstrates that future transmission rates may be predicted reasonably well during the limited experiments conducted. However, the algorithm should be evaluated thoroughly to determine whether it performs sufficiently for a much wider set of representative circumstances.

Chapter 8

On Regulating Traffic in Shared Wireless Access Media

In the previous chapter, a traffic control solution is proposed that relies on the basics of the performance model from Chapter 2. Based on a multi-service traffic profile, a QoS budget may be determined and assigned to each terminal in a wireless network to guarantee the QoS for the given set of application sessions. Using this method, the terminal can determine locally if the network consumption does not exceed the assigned budget and whether a new application session may fit within the QoS budget given.

However, when a terminal exceeds the assigned QoS budget, the QoS guarantees of other users may not be met. To this end, traffic shaping and policing mechanisms can be used to make sure that the total amount of traffic stays within the budget. In Chapter 7 the traffic in the upstream direction is distinguished from the downstream traffic for several types of applications and quality levels.

In shared wireless access media the uplink and downlink use the same frequency bands and therefore compete for the medium when transmitting packets. This occurs in various types of practical wireless communication systems and leads to an inefficient medium utilization when a traffic contract maintains a strict separation between the two directions.

In this chapter a method [116] for traffic policing is introduced and evaluated that uses shared communication media more efficiently by exchanging the contract parameters between the uplink and the downlink directions dynamically, when desired. The solution is based upon a commonly used policing mechanism. An extensive performance study shows how the mechanism should be configured to use it effectively.

This chapter is based on the results presented in [49] and [92].

8.1 Introduction

To realize quality of service in communication networks, end-users establish a traffic contract with the network based on the quality of experience desired at the application level. Common parameters to characterize a traffic flow include the peak data rate, average sustained data rate, and the burst size. Checking traffic rates and comparing them with the traffic contracts is referred to as traffic policing. If the rate exceeds the contract, packets could be marked or dropped, depending on the local policy. Using the traffic control method described in Chapter 7, a QoS budget is determined that can subsequently be used for traffic policing. Traditionally, only traffic that flows into the network was subjected to a predefined service level. The rationale for this is that if a network is properly dimensioned and the sum of all service level agreements does not exceed the network capacity, traffic will smoothly propagate through the network and hence checking inflow traffic for compliancy with the agreements is sufficient. Traffic contracts for Internet access and for interconnecting enterprises often dictate rates in both the up-and the downstream directions.

The token bucket was first mentioned in [113] and has ever since been a common approach to police traffic contracts. It operates with two parameters: the token rate (TR) and the bucket limit C. Every $1/TR$ seconds a token is added to a bucket with limit C. Tokens are discarded if the bucket is full. When a packet of s bytes arrives, s tokens are removed from the bucket and the packet can pass. If the bucket contains fewer than s tokens, no tokens are removed from the bucket and the packet is considered as non-compliant. The algorithm allows bursts of up to C bytes, but over the long run the output of conforming packets is limited to the constant rate TR. Non-compliant packets can be dropped, queued or marked, depending on the policy. A commonly used policy is instructing network elements to drop marked packets (randomly) in case of overload situations.

Traffic contract parameters are negotiated on a relatively long time scale (in terms of weeks or months) and today's traffic policing techniques cannot adaptively change contract limits of the uplink and/or the downlink. In wired systems, the up- and the downstream connections often have a separate physical connection, making separate traffic contracts a necessity. In shared media as e.g., WLAN, where the sending of information in the upstream direction competes with the sending in the downstream direction, policing in the two directions independently in general degrades the efficiency of a shared communication medium. As an example, consider the situation that the network has more information to send to a tagged client over the downlink than the downlink service contract allows. At the same time, assume that the client transmits less than the agreed uplink traffic contract sustains. Applying the conventional traffic policing techniques to the uplink and downlink directions independently may queue or drop surplus downlink traffic, even though there may be room in the uplink contract that uses the very same medium for transportation. By considering the total service contract in the two directions simultaneously, vari-

ations in the demand for the downlink can be compensated with temporarily sending less information in the upstream direction, and vice versa. This cannot be realized by applying the traditional token bucket policing mechanisms, where each direction has its own bucket, which operates autonomously.

The policing of a traffic contract goes hand in hand with the dimensioning of the network resources reserved for all traffic contracts. Ross discusses in [107] various models for computing the reservation level of a trunk that is shared by various classes of traffic which may differ in their demand and/or mean holding times. These models operate on relatively long time scales and provide insight in blocking probabilities of (state-dependent) trunk reservation schemes. Borst and Mitra describe in [23] the virtual partitioning scheme, introducing dynamics to these reservation schemes by giving lower priority to misbehaving classes which violate their predetermined capacity allocation. Their outcome also operates on a long time scale and provides fairness and robustness in the admission of new session initiations, giving lower priority to misbehaving classes. The policing mechanism proposed in this chapter can serve as the basis for the virtual partitioning scheme and for alternative resource reservation approaches. Both need one another to make traffic contracts effective.

This chapter introduces and evaluates a method [116] that polices traffic in shared communication media. This is realized by treating the traffic contract as a whole and distributing it dynamically over the up- and downstream directions, when needed. By considering the total service contract in the two directions simultaneously, variations in the demand for the downlink can be compensated with temporarily sending less information in the upstream direction, and vice versa. An extensive performance analysis demonstrates the gains achieved by applying this new mechanism and provides insight in the side conditions when the solution can be applied best. Moreover, this performance analysis shows how the parameters of the mechanism should be chosen such that good performance is achieved, oscillation effects are prevented and traffic in either the upstream or the directions cannot dominate the total contract.

8.2 Mechanism for policing traffic in shared media

Policing in the up- and the downstream directions simultaneously can be realized by two independent policing entities that read from a common token reservoir. The rate at which the reservoir is filled represents the total traffic contract, namely the sum of the rates of both directions. However, special precautions are needed to prevent that one of the directions dominates the traffic contract and hence one direction suffocates in favor of the other. Starvation of one direction eventually results in disconnection (or total uselessness) of a bilateral connection. Prioritizing one direction to solve this for data applications is tempting. After all, packets for the downstream direction may have traveled from far and have nearly reached the final destination and hence giving them

priority over upstream packets may seem a good idea at first glance. However, in the case of data applications, the upstream direction consists of mainly acknowledgements of received packets which are small in size by nature and dropping them will disturb the self-clocking mechanism of TCP, as described by Jacobson [63], and causes it to lower its throughput and resend information that will eventually need to pass the downstream access link. For UDP-based real-time applications, the connections in both directions often contribute equally to the quality experienced by the person using the real-time application. The aim is a policing mechanism that improves the overall throughput of the shared wireless medium by taking the sum of the traffic contract of the up- and downstream directions, and at the same time prevents starvation of one or both of the directions. An important prerequisite is that the solution should minimize changes required on existing equipment that implements the token bucket mechanism.

8.2.1 Mechanism to police in upstream and downstream directions concurrently

Consider two regular token bucket policing mechanisms: one in the upstream and one in the downstream direction, named U and D, respectively. Without loss of generality, the remainder assumes that U and D may also be used for spacing and hence contain a queue to temporarily store non-compliant packets. The mechanisms U and D regulate traffic of a terminal named T. The mechanism can also be used to regulate a group of terminals (or an interface) but for understanding to the reader the description concentrates at the situation where only one terminal feeds the mechanisms U and D. Several of these policing mechanisms may work in parallel to police various groups of terminals simultaneously but since each of them can work autonomously, describing one pair $(U + D)$ for policing a single terminal is sufficient.

Updates of the policing mechanisms parameters are only needed when conformance is checked, i.e., upon the arrival of a packet in either the upstream or the downstream direction. Figure 8.1 illustrates the state diagram. The index $i(i = 0, 1 \ mod \ 2)$ indicates the direction (e.g. $i = 0$ for the upstream direction and $i = 1$ for the downstream direction). Let L_i denote the bucket level of direction i. Policing mechanisms U and D must support two bucket limits, namely a ceiling bucket limit C_i and a floor bucket limit F_i. The ceiling limits C_1 and C_2 limit the burst of packets for each direction, whereas the floor limits F_1 and F_2 prevent that one direction suffocates in favor of the other.

When a packet of size s arrives for transmission in direction i, first the token level L_i of the policing mechanism for the corresponding direction is checked. If the token level stores sufficient tokens to let the packet pass, the level is reduced according to the size of the packet, just as regular leaky bucket systems commonly used today. If, however, the token level is less than s, the size of the arriving packet, the conclusion that the packet is non-conformant could be prevented by using tokens from the bucket of the other direction (i.e., $i + 1 \ mod$

2). This realizes the shared token consumption.

To prevent that one direction dominates the overall traffic contract and hence suffocates the other direction, a floor limit is introduced that limits the usage of tokens from the other direction: direction i can use tokens from the reservoir of direction $i+1$ unless the token level L_{i+1} drops below the floor threshold F_{i+1}. A packet is marked as non-conforming if both the bucket limit L_i is lower than zero and the bucket limit of the other direction L_{i+1} is below the threshold value F_{i+1}, when reduced with s the size of the arriving packet. The ceiling bucket limit denotes the maximum level of the reservoir, similar as in traditional token bucket systems.

To realize the sharing of the reservoir in good times and bad times, the waste of tokens as a result of overflowing during low traffic activities in direction i can be added to L_{i+1}. The parameters C and F can be chosen such that an optimum balance exists between performance, the maximum burst size of consecutive packets and the tolerance level that one direction dominates the total traffic contract. Note that by selecting $F_i = C_i$ for both directions $i = 0, 1$, the token bucket mechanism as used today is obtained. A natural location for placing a policing mechanism in a commercial WLAN architecture environment is at the WLAN controller. This location is preferred over the AP, mainly since it prevents loosing the state space of policers when a station roams from one AP to another. In the case residential WLAN access points are used to offer access to the public, the mobility broker is a good location for the policing mechanism. The mobility broker plays an important role in realizing handovers between residences that are connected to different network providers and checks traffic contracts, see [94].

By placing mechanisms U and D in the same functional architectural unit, exchanging information can be realized effectively through e.g., socket communication. Note that the implementation of the mechanism does not require the exchange of state space between U and D. Information exchange between the up- and downstream policing mechanisms can be limited to: 1) the number of surplus tokens transferred to reservoir $i+1$ when the ceiling limit $C(i)$ is exceeded, and 2) the request whether $s - L(i)$ tokens can be subtracted from reservoir $i+1$. A response to the first type is not necessary. If the response to the second type of information exchange is positive, the token level of reservoir $i+1$ is reduced by $s - L(i)$ and the token level of reservoir i is set to zero. If the response is either negative or not received, the packet should be considered as non-compliant, which only affects the state space of the initiating policing mechanism. Hence, synchronization of mechanisms U and D is no necessary prerequisite.

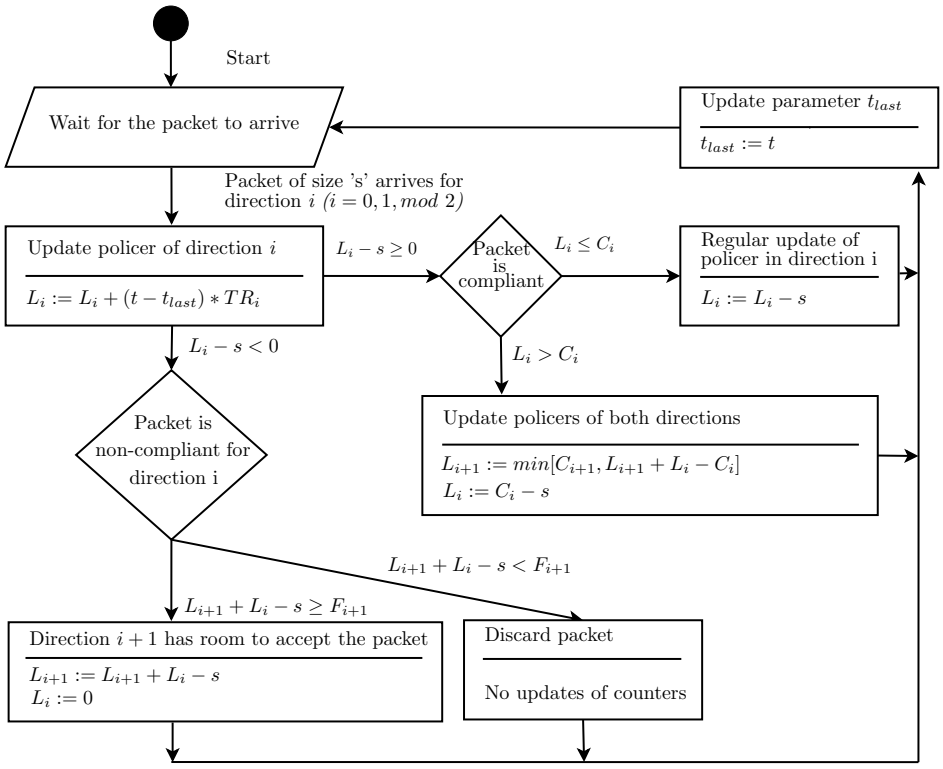


Figure 8.1: State diagram for policing in two directions concurrently and preventing that one of the directions dominate the traffic contract. Here the parameters L_i, C_i, F_i, TR_i denote respectively the bucket limit, the ceiling bucket limit, the floor bucket limit, and the token rate in direction i ($i = 0, 1\ mod\ 2$).

8.2.2 Example

Let $i = 0$ denote the upstream direction and $i = 1$ the downstream direction. Figure 8.2 depicts the token level of the U and D token bucket mechanisms over time, the upstream bucket U located at the top and the downstream bucket D at the bottom. At time $T = t_1$, a packet with size s_1 arrives for transmission in the upstream direction. Since the value s_1 can be subtracted from the token level of the upstream reservoir U, there is no need for any interaction with the downstream bucket D. At time t_2, a packet arrives at the downstream bucket. As the packet arrives after the ceiling limit of the downstream bucket has been reached, an amount of a_2 tokens remains unused by the downstream bucket and is transferred to the upstream level. In this case, the upstream bucket limit benefits from the low activity in the downstream direction.

At times $T = t_4$, $T = t_5$, and $T = t_6$, downstream packets arrive. After subtraction of the packet sizes s_4, s_5 and s_6 a negative bucket limit value of respectively a_4, a_5 and a_6 would occur at the downstream policer D. To prevent marking these packets as non-compliant, the token level of the upstream bucket is reduced with these values. This is allowed as these decreases do not realize a token level that lies below the threshold value F_0, the floor limit of policer U.

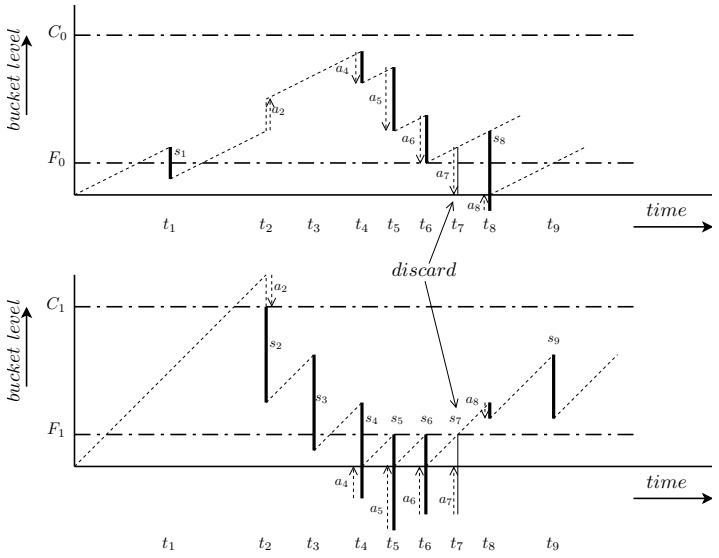


Figure 8.2: Example of how the mechanism that polices traffic in the up- and downstream directions concurrently. Index $i = 0$ denotes the upstream direction, with corresponding C_0 and F_0 as the ceiling and floor limits. Index $i = 1$ denotes the downstream direction.

The packet that arrives in the downstream direction at time $T = t_7$ is considered non-compliant, since subtracting the value a_7 (the deficit after subtracting the value s_7 from the downstream token level at $T = t_7$) from the upstream tokens will result in lowering the upstream token level below the floor threshold level F_0. The packet arriving at $T = t_8$ in the upstream direction is compliant again, as at $T = t_8$ the downstream token level L_1 is more than the value a_8 above the floor value F_1.

8.3 Performance evaluation

By flexibly taking and assigning tokens among both directions, intuitively the shared medium is used better than when separating the contract over the two distinct directions. The analysis of this section shows the effects of using the introduced policing mechanism quantitatively. The performance evaluation is performed analytically and by simulation. Results obtained from simulations provide insight in the performance at the application layer and quantify the improvement that can be expected in practical situations.

8.3.1 Analytical evaluation

In the analytical evaluation the exchange of tokens between the up-and downstream bucket is modeled and evaluated. The outcomes indicate how efficiently the tokens are used as a measure on how efficiently network resources (if available) may be utilized.

Mathematical model

Consider a leaky bucket operating in a slotted system that contains two kinds of tokens: tokens for the upstream direction can be distinguished from tokens for the downstream direction (e.g., by means of colors). The input rates for both the up- and the downstream directions are modeled by incrementing the number of tokens by one for each direction, at each time slot. To model the bucket's leak rate, each time slot a packet for the up- and/or the downstream directions may arrive mutually independent from one another with probability P_u and P_d, respectively.

Upon arrival of a new packet, the token level of the corresponding direction decreases with the size of the packet, if there are sufficient tokens. Note that this system models a leaky bucket and not a wireless transmission system and hence at each time slot a packet may arrive in either or both the upstream and the downstream direction. Assume that K_i, the size of packets for direction i, is geometrically distributed with mean $1/(1 - a_i)$ units $(0 < a_i < 1)$, i.e.,

$$Pr(K_i = k) = (1 - a_i) \cdot {a_i}^{k-1} \quad (k = 1, 2, \ldots). \tag{8.1}$$

Let X_u and X_d denote the number of tokens in de bucket of respectively the upstream and the downstream direction. The tuple (X_u, X_d) then forms a

discrete-time Markov chain, where upon packet arrival we choose to first add the input rates to the bucket and before lowering the token level with the packet size. If the token level is sufficiently high, taking from the other direction is allowed, which requires keeping track of the number of up- and downstream tokens in the bucket.

If no packets arrive (or the size of the packets exceeds the size of the individual token levels), the transition $(x_u, x_d) \rightarrow (x_u + 1, x_d + 1)$ occurs. Two packets, one in each direction, arrive with size $s_u(> 0)$ and $s_d(> 0)$ with probability $p_u \cdot Pr(K_u = s_u) \cdot p_d \cdot Pr(K_d = s_d)$. In that case, the bucket is decreased with $s_u + 1$ (and/or or $s_d + 1$) tokens of the corresponding direction. Consequently, the transition $(x_u, x_d) \rightarrow (x_u + 1 - s_u, x_d + 1 - s_d)$ occurs, if and only if $s_u \leqslant x_u + 1$ and $s_d \leqslant x_d + 1$. If one of these conditions do not hold, the upstream direction may grab tokens from the downstream direction (and vice versa), if $x_d > s_d + s_u - x_u - 2 + F_d$, where F_d denotes the floor limit of the downstream direction. Note that the adding of 1 to the levels x_u and x_d stems from the input rate at every time slot and only takes place if $x_u \neq C_u$ and $x_d \neq C_d$, where C_i denotes the ceiling limit of the bucket for direction i.

After taking from the other direction, the token level of the direction that uses is reduced to zero. Upon arrival of a packet, the token level of the corresponding direction is lowered first before the other direction can take tokens. This means that if the size of an arriving packet g exceeds the token level of its direction as well as the token level of the other direction minus its floor value (after this level was possibly reduced with the size of the packet that may have arrived in that direction), the packet g will be discarded and the token level corresponding with the direction of g will be increased with one unit, if it has not reached its ceiling yet. The state space Ω of the number of up and down tokens in the bucket is expressed by $\Omega = \{(x_u, x_d) = (i, j) | i = 0 \cdots C_u, j = 0 \cdots C_d\}$. By putting all $|\Omega| = (C_u + 1) \cdot (C_d + 1)$ transitions in a matrix M, this matrix M forms the transition matrix of a discrete-time Markov chain and hence the stationary distribution π of (X_u, X_d) can be determined by computing the left eigenvector of M for the eigenvalue of 1.

The probability that the down direction grabs tokens from the upstream direction and the other way around can be used as a measure to determine whether one direction suffocates by favoring the other. These grabbing probabilities are denoted as $P(DU)$ and $P(UD)$, respectively. After computing the stationary distribution of the number of tokens in each direction $\pi(x_u, x_d)$, the probability $P(DU)$ can be computed by considering each transition that the Markov chain enters the area where the down direction borrows from the up direction, given

the number of up and down tokens in the bucket at that moment.

$$
\begin{aligned}
P(DU) &= \sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d} P(DU|(X_u=x_u, X_d=x_d))\cdot\pi(x_u,x_d) \qquad (8.2)\\
&= \sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Bigg\{(1-p_u)\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j=0)} p_d\cdot\\
&\cdot\quad P(K_d=i)+\sum_{j=1}^{j_{\text{end}}}\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j)} p_u P(K_u=j)\cdot p_d P(K_d=i)\Bigg\}\\
&= (1-a_d)p_d\cdot\sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Bigg\{(1-p_u)\cdot\\
&\cdot\quad \sum_{i=i_{\text{start}}}^{i_{\text{end}}(j=0)} a_d^{i-1}+p_u(1-a_u)\cdot\sum_{j=1}^{j_{\text{end}}}\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j)} a_d^{i-1}a_u^{j-1}\Bigg\}\\
&= p_d\cdot\sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Big\{a_d^{i_{\text{start}}-1}\cdot(1-p_u a_u^{j_{\text{end}}})+\\
&- \quad a_d^{i_{\text{end}}(j=0)}\left\{1-p_u+\frac{p_u(1-a_u)\cdot(1-(\frac{a_u}{a_d})^{j_{\text{end}}})}{a_d-a_u}\right\},
\end{aligned}
$$

where parameters $j_{\text{end}} = x_u+1-F_u-\sigma(x_u=C_u)$, $i_{\text{start}} = x_d+2-\sigma(x_d=C_d)$, $i_{\text{end}}(j) = x_d+1-j-\sigma(x_d=C_d)+\beta$, and $\beta = (x_u+1-\sigma(x_u=C_u)-F_u)\cdot\sigma(\min(x_u+1,C_u)>F_u)$. The function $\delta(e)=1$ if $e=true$ and equals zero in the case $e=false$.

Similarly for the case that the upstream direction grabs tokens from the downstream direction $P(UD)$:

$$
\begin{aligned}
P(UD) &= \sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d} P(UD|(X_u=x_u, X_d=x_d))\cdot\pi(x_u,x_d) \qquad (8.3)\\
&= \sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Bigg\{(1-p_d)\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j=0)} p_u\cdot\\
&\cdot\quad P(K_u=i)+\sum_{j=1}^{j_{\text{end}}}\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j)} p_d P(K_d=j)\cdot p_u P(K_u=i)\Bigg\}\\
&= (1-a_u)p_u\cdot\sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Bigg\{(1-p_d)\cdot\\
&\cdot\quad \sum_{i=i_{\text{start}}}^{i_{\text{end}}(j=0)} a_u^{i-1}+p_d(1-a_d)\cdot\sum_{j=1}^{j_{\text{end}}}\sum_{i=i_{\text{start}}}^{i_{\text{end}}(j)} a_u^{i-1}a_d^{j-1}\Bigg\}\\
&= p_u\cdot\sum_{x_u=0}^{C_u}\sum_{x_d=0}^{C_d}\pi(x_u,x_d)\cdot\Big\{a_u^{i_{\text{start}}-1}\cdot(1-p_d a_d^{j_{\text{end}}})+\\
&- \quad a_u^{i_{\text{end}}(j=0)}\left\{1-p_d+\frac{p_d(1-a_d)\cdot(1-(\frac{a_d}{a_u})^{j_{\text{end}}})}{a_u-a_d}\right\},
\end{aligned}
$$

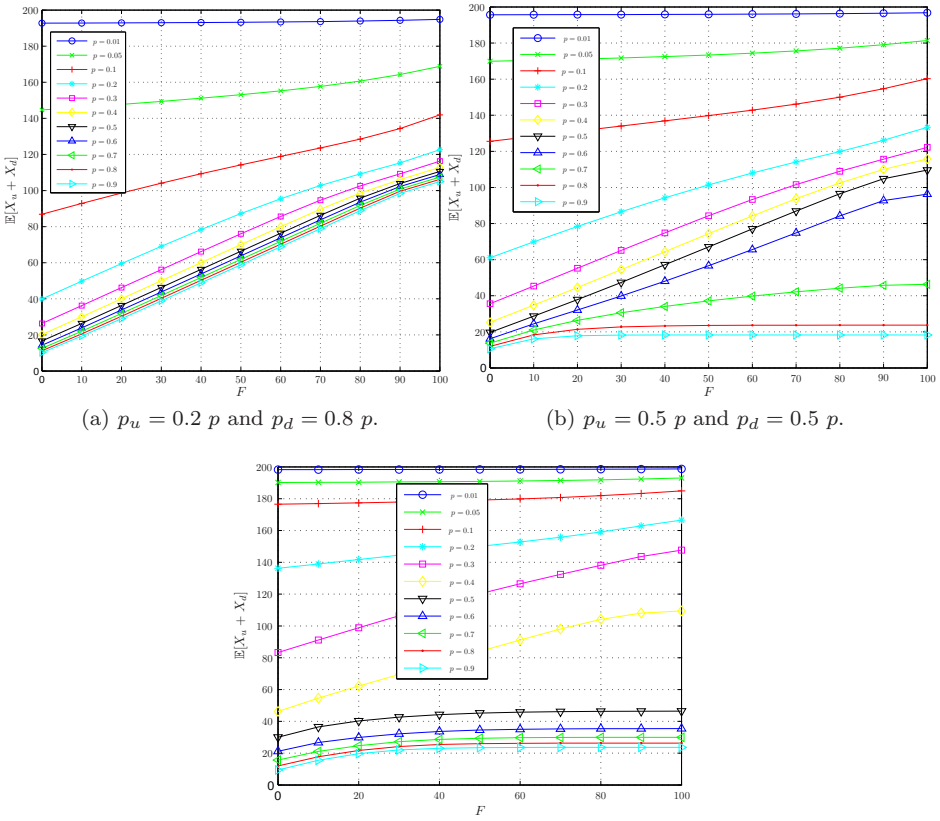
where $j_{\text{end}} = x_d + 1 - F_d - \sigma(x_d = C_d)$, $i_{\text{start}} = x_u + 2 - \sigma(x_u = C_u)$, $i_{\text{end}}(j) = x_u + 1 - j - \sigma(x_u = C_u) + \beta$, and $\beta = (x_d + 1 - \sigma(x_d = C_d) - F_d)\cdot\sigma(\min(x_d+1, C_d) > F_d)$.

Evaluation through the mathematical model

To compute the performance measures of the mathematical model, the mean packet size of the upstream packet was set to three units whereas the mean size of the downstream packets was 30 units ($a_u = 2/3$, $a_d = 29/30$). The state space Ω of the Markov chain explodes rapidly if the ceiling limits become large. The ceiling limit of the bucket equals 100 units for each direction ($C_u = C_d = 100$). The floor limits F_U and F_D are always equal to one another ($F = F_u = F_d$) and are increased in steps of 10% of the corresponding ceiling limit.

The parameter p was introduced to model the load of the medium and the values for p_u and p_d were deducted from this parameter by multiplying with values $0.2 - 0.8$, $0.5 - 0.5$, $0.8 - 0.2$, resembling the case that traffic rate in the downstream direction is dominant, traffic load in the downstream and upstream directions contribute equally to the load and finally the case that the intensity

of packets for the upstream direction dominates the traffic load.



(a) $p_u = 0.2\ p$ and $p_d = 0.8\ p$.

(b) $p_u = 0.5\ p$ and $p_d = 0.5\ p$.

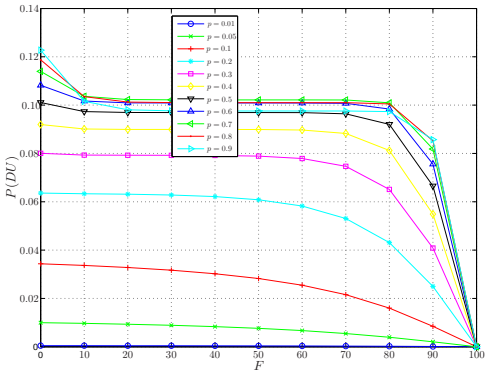
(c) $p_u = 0.8\ p$ and $p_d = 0.2\ p$.

Figure 8.3: Mean bucket size, $\mathbb{E}[X_u + X_d]$, as a function of the floor value, F, for different load conditions p. In In Figure 8.3a the intensity of downstream packets dominates the traffic load, in Figure 8.3b the downstream and upstream packet equally contribute to the load and in Figure 8.3c the intensity of upstream packets dominates the traffic load.

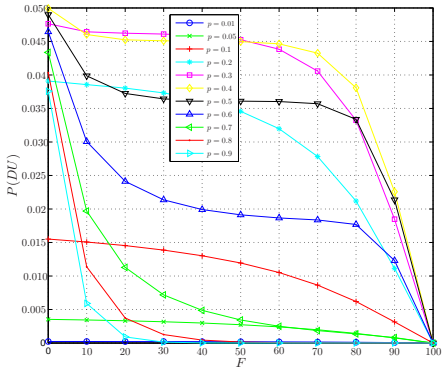
Figures 8.3a to 8.3c depict the mean number of tokens in the bucket (up + down), denoted $\mathbb{E}[X_u + X_d]$, as a function of the floor limits $F = F_u = F_d$ for different load conditions. The figures show that $\mathbb{E}[X_u + X_d]$ increases if the floor values increase. This confirms the initial thought that the mechanism that polices in two directions concurrently uses the shared medium more effectively: the less tokens in de bucket, the more packets can be sent over the medium. The question whether the wireless and/or fixed media can support the high load

values of p is subject of study for admission control and hence irrelevant in this context. Recall that the floor limit $F = 100$ corresponds with the traditional leaky bucket policing mechanism. The gain corresponding to a floor limit F can be calculated by subtracting the mean token level for that floor limit from the mean token level for floor limit $F = 100$ (no mutual sharing of resources).

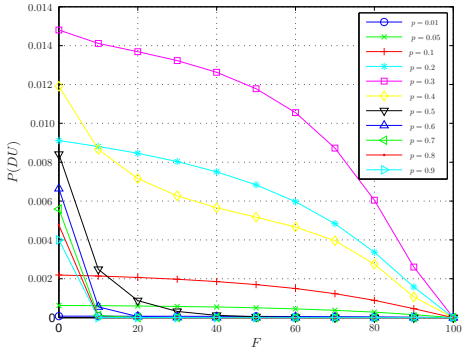
Figures 8.3a to 8.3c show that for very low loads the sharing of resources between directions will hardly take place and hence the gain of applying the new policing mechanism is hardly noticeable. If larger downstream packets dominate the total traffic volume (Figure 8.3a), the slope of the curves are steepest for the highest load conditions, revealing more gain when applying the mechanism that polices in two directions concurrently. In the case the load on the medium is equally distributed over packets in the up- and downstream directions (Figure 8.3b), the slope of the curves become flat for very low and very high loads. This effect becomes stronger if the smaller upstream packets dominate the network traffic (Figure 8.3c), revealing a maximum in the performance gain for medium load conditions.



(a) $p_u = 0.2\ p$ and $p_d = 0.8\ p$.



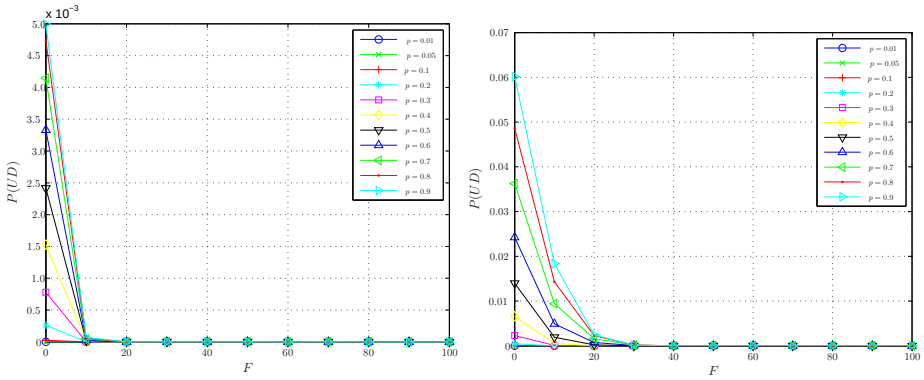
(b) $p_u = 0.5\ p$ and $p_d = 0.5\ p$.



(c) $p_u = 0.8\ p$ and $p_d = 0.2\ p$.

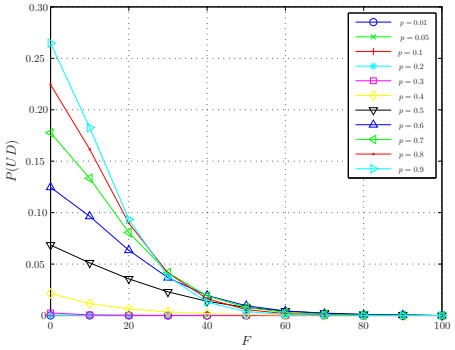
Figure 8.4: The probability, $P(DU)$, that the downstream direction borrows from the upstream direction as a function of the floor value, F, for different load conditions p. In Figure 8.4a the intensity of downstream packets dominates the traffic load, in Figure 8.4b the downstream and upstream packet equally contribute to the load and in Figure 8.4c the intensity of upstream packets dominates the traffic load.

Figures 8.4a to 8.4c depict for different load conditions the probability that the downstream direction borrows from the upstream direction, $P(DU)$, as a function of the floor limit F. Note that for $F = 100$, the floor limit coincides with the ceiling, resulting in zero probability to grab tokens from the up direction, regardless the traffic load. In the case the traffic rate in the downstream dominates (Figure 8.4a), the probability $P(DU)$ declines slowly as a function of the floor limit. Similar observations can be made when the intensity of packet arrivals for the uplink becomes equal or even higher than the intensity packets for the downlink arrive, see Figures 8.4b and 8.4c. For the slowly declining conditions, the grabbing probability hardly changes if the floor limit F is in the range of $20\% - 70\%$ of the ceiling limit.



(a) $p_u = 0.2\ p$ and $p_d = 0.8\ p$.

(b) $p_u = 0.5\ p$ and $p_d = 0.5\ p$.



(c) $p_u = 0.8\ p$ and $p_d = 0.2\ p$.

Figure 8.5: The probability, $P(UD)$, that the upstream direction borrows from the downstream direction as a function of the floor value, F, for different load conditions p. In Figure 8.4a the intensity of downstream packets dominates the traffic load, in Figure 8.4b the downstream and upstream packet equally contribute to the load and in Figure 8.4c the intensity of upstream packets dominates the traffic load.

Figures 8.5a to 8.5c depict for different load conditions the probability that the upstream direction grabs tokens from the downstream direction, $P(DU)$, as a function of the floor limit, F. Since the size of upstream packets is only 10% of the size of downstream packets, the grabbing probability $P(UD)$ is low in the case the downstream direction dominates the traffic rate. The borrowing probability declines more slowly as a function of the floor limit if the contribution of the upstream packets to the total traffic rate increases over the relative contribution of the downstream direction (approx. 10^{-3}, not shown). If the upstream packets dominate the traffic rate, the borrowing probability is roughly reduced with half its size if the floor limit equals 20%, when compared to the borrowing probability for full sharing (i.e., $F = 0$).

8.3.2 Simulation study

This section studies by means of simulation whether or not higher medium efficiencies can be achieved in circumstances that may occur in practice. To this end, a simulation study is conducted to gain insight in the absolute performance gains of the proposed policing mechanism over the traditional policing mechanism that specify separate traffic volumes for up-and downstream traffic.

In general, two telecommunication traffic types can be distinguished, namely elastic (TCP-type, e.g., FTP, HTTP data) and real-time (e.g., voice / video kind of applications that use the unreliable UDP layer). For real-time traffic, the performance changes as a result of using the new policing mechanism can best be expressed by the perceived user experience (measured by opinion scores) and is beyond the scope of this work. Elastic traffic, however, has built-in congestion avoidance functionality that effectively shares the transmission capacity equally over all active flows. Blocking TCP packets as a result of non-compliancy of the traffic contract inevitably results in retransmissions and hence increases the duration of the associated service as experienced by the end-user. Hence, the length of the download time serves as a good basis to quantify the affect of the decision to either block or let a packet pass.

Browsing application characteristics

In our simulation study a number of wireless users browse the Internet of which the page statistics are based upon a set of telecommunication companies and research institutes, shown in Table 7.3. The browsing application uses these values to generate a web page with a randomly generated body size and a random number of inline objects with a random size. These three random variables are each drawn from a uniform distribution that matches the respective measured parameter range. As a result, the average page size retrieved in our simulation is 131270 bytes.

With regard to modelling respresentative browsing behavior of the wireless users, the time between the consecutive clicks to retrieve a web page in a user's browsing session was drawn from a lognormal distribution function with a mean and standard deviation of 36.8 and 56.4 seconds respectively, see [78].

Network infrastructure

The considered network infrastructure in which the wireless users expose their browsing behavior is depicted in Figure 7.2. In this chapter we assume that the users retrieve their web pages through an IEEE 802.11g-based WLAN access point via an Internet backbone from an FTP server that is located on the Internet. The WLAN AP and the FTP server are each connected by FastEthernet links to an IP backbone router. Optical OC-48 links interconnect our backbone routers with packet-level characteristics that match a U.S. transcontinental connection.

A number of previous efforts have measured Internet backbones to study the packet-level characteristics. In [83], the authors are particularly interested in real-time applications and have performed active measurements (by injecting traffic) over paths that belong to seven different U.S. Internet providers. Others have passively observed and recorded traffic to characterize the packet-level characteristics of long-distance connections [38, 65, 66] using the IP Monitoring (IPMON) system on the OC-48 links in the Sprint IP backbone network. These studies generally conclude that the mean path delay and jitter on long-distance links is relatively small, whereas the packet loss is very small in general.

In fact, the results presented in [38] reveal that the mean delay on various U.S. transcontinental TCP connections is in general nearly constant and the amount of jitter is insufficient to impact the performance of even delay constrained-applications. Furthermore, the measurements indicate that most TCP flows experience no end-to-end loss. More specifically, from the 2×10^8 recorded packets that have crossed five POPs (Points of Presence) and eight core routers between San Jose and New York the obtained delay distributions indicate that in both directions the delay is nearly constant at 28 ms.

In our simulation study we assume the presence of a U.S. transcontinental link with similar bidirectional delay values as the one observed in [38] between San Jose and New York. To this end we have configured the link between our backbone routers to impose a constant delay of 28 ms to packets without introducing any loss. The network nodes (user stations and web server) are configured with a TCP stack that matches the implementation of the Windows XP operating system, which applies the Selective ACK (SACK) mechanism that is used to reduce the impact of TCP retransmissions. The proposed policing mechanism is embedded in the WLAN AP to police the up-and downstream traffic in accordance with a pre-defined traffic contract using two configurable buckets, each parameterized by a bucket limit and floor limit, as outlined in Section 8.2.

Traffic contract

The next step in evaluating the effectiveness of the proposed policing solution is to define a traffic contract, based on the quality of experience desired at the application level. In this study we quantify the benefit of the presented traffic policing mechanism in terms of page response time, being the time needed to transfer and process the entire webpage from the sending to the receiving side.

The maximum quality of experience delivered in our considered network for the given user behavior and web page characteristics is reached when no traffic policing is applied. The associated traffic volume per time-unit defines the network resource requirements in up-and downstream direction to deliver the best application performance (i.e., lowest mean page response times). In practice, connection acceptance mechanisms may account for pre-defined, application-

specific quality of service levels and associated traffic volumes [93]. Alternatively, it may be up to the users themselves to accommodate their applications within the total traffic contract. In the former case, application-specific traffic requirements, and thus their characteristics, may be known when admitted to the network. Whereas in the latter case, there is no 'advance knowledge' on the exact network resource requirements of the applications.

In our simulation study we assume the possibility of these two cases. In the advance knowledge case, the traffic contract corresponds to the network resource requirements in up-and downstream in the same ratio as for the unpoliced traffic volumes. To this end, the browsing application is first simulated without any traffic regulation restrictions. This yields a benchmark, used to compare the performance of the new policing mechanism.

To determine the resource requirements without traffic regulation restrictions, the browsing application characteristics and network infrastructure are modeled in the OPNET Modeler network simulation tool, version 15.0. Simulation runs of twenty days (simulation time) were performed to yield accurate outcomes from approximately 47×10^3 page downloads with an average page response time of 0.94 seconds, requiring (at the IP layer) on average 335 and 4000 bytes-per-second in the up-and downstream direction respectively. These values are subsequently used as the token rates for the up- and downstream leaky bucket traffic policing parameters.

In the case of symmetric traffic contracts: 50% of the total token rate for the upstream and 50% for the downstream direction. The browsing application was extensively simulated for various bucket parameters and both the symmetric and 'advanced knowledge' traffic contracts.

To understand the effect of sharing the traffic contract among various traffic streams while eliminating delay effects as a result of the WLAN protocol (e.g., due to the back off period), simulations were performed for different user populations that share a single contract, namely one, five and ten users. Each user downloads a web page according to the pre-defined statistics while the token rate was chosen proportionally with the user population. The mean page response times were obtained from consecutive simulations while the token rate was reduced in equal steps of 10% of the original contract, starting at 100% and stopping at 10%. To study the effect on the tolerance of the policing mechanism, the simulations were repeated for various conditions of the token bucket limit, varying this limit in steps of $10s$, from $10s$ to $60s$ and additionally for $90s$, $120s$, and for $60000s$. In our simulations, the bucket limits were expressed in bytes, obtained by multiplying the bucket limit in seconds with the used token rate in bytes per second. The floor limits of the up- and downstream buckets (F_u and F_d, respectively) that indicate to which extent the mutual borrowing of tokens is tolerated may be set in different ways: (1) static, as a percentage of the (fixed) bucket limit C_i or (2) dynamic, as a percentage of the actual number

of tokens L_i in each bucket. In the first case, which we refer to as *static floor limit setting*, token exchange is blocked when there are less tokens than F_i in the bucket. In the latter case, direction i can always borrow from the other direction $i+1$ *mod* 2 as long as $F_i + 1 < 100\% \times L_i + 1$, named *dynamic floor limit setting*. Both systems behave identically when $F_i \in \{0\% \times L_i, 100\% \times L_i\}$. In the remainder $F_i = x\% \times L_i$ is denoted $F_i = x\%$.

Note that $F_i = 0\%$ corresponds to a complete sharing situation, meaning that the up- and downstream buffers can always grab tokens from one another, if available. The $F_i = 100\%$ floor limit corresponds to the situation that the two buffers cannot exchange tokens and hence coincides with the traditional way of policing traffic.

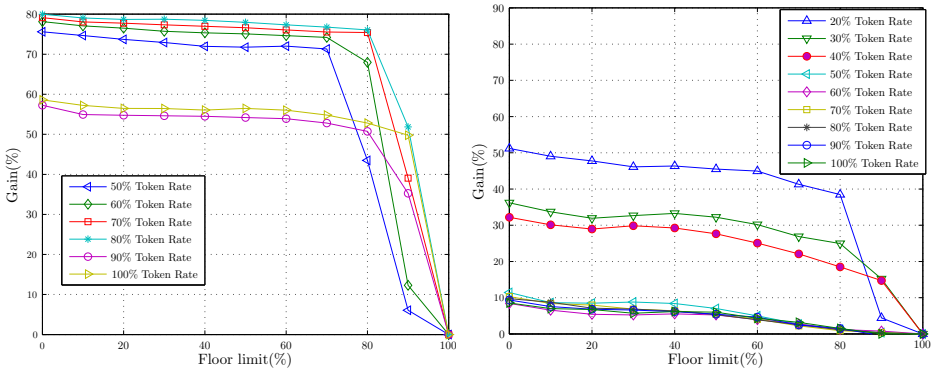
In order to evaluate the effectiveness of the proposed policing solution quantitatively for a broad set of parameters, over 10.000 simulation runs of 20 days each were conducted. The quantitative comparison of the new mechanism is made by comparing the benchmark results (the simulated mean page response times of the traditional policer) ($\overline{R}_{F_{100\%}}$) against the mean page response times from new policing mechanism, for various floor limits. The improvement, or gain, in the mean page response time is determined as follows: $gain = (\overline{R}_{F_{100\%}} - \overline{R}_{F_i}) \cdot 100\%/\overline{R}_{F_{100\%}}$, where $F_i = 0\%, 20\%, \ldots, 100\%$.

8.3.3 Advanced knowledge results

In Figure 8.6a the improvement in mean page response time is shown for the policing solution with $F_i \neq 100\%$ over the systems for a static floor limit setting of $F_i = 100\%$ and bucket limits $C = C_0 = C_1 = 10$ seconds. The graph indicates that the new policing mechanism performs better when the traffic contract is significantly reduced, realized by a low token rate. If the token rate is chosen too small, TCP connections time-out and pages cannot be retrieved for higher floor limit values. This underlines the superior performance of concurrent policing ($F_i = 0\%$ in particular) over traditional policing ($F_i = 100\%$) and explains why Figure 8.6a displays no data for very small service contracts (40% of the original token rate and less) and high floor limits.

The newly developed policing mechanism clearly suffers less from TCP time-outs and pages could even be retrieved for token rates set to 10% of the original traffic contract (naturally with high mean page response times). When increasing the bucket limit from ten to thirty seconds (see Figure 8.6b), the improvement is smaller, but still significant for small traffic contracts and modest, consistent gains for larger ones. This can be explained by recalling that during the time between consecutive clicks to retrieve a web page, the token buffer is filled to the top. If the bucket limit increases, more tokens are available during periods of scarcity. Another reason for the smaller gains is that there are fewer tokens exchanged because the floor limit is set statically to a percentage of the bucket limit and thus raises the threshold for exchanging tokens when increasing the

bucket limit.

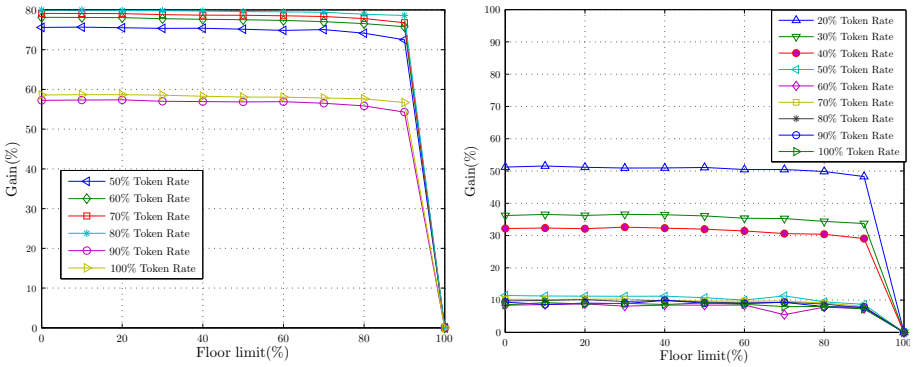


(a) $C_1 = C_2 = 10$s. (b) $C_1 = C_2 = 30$s.

Figure 8.6: Improvements in mean page response times for a bucket limit of 10 and 30 seconds. Comparing the policing solution for $F_i \neq 100\%$ of C_i with a system where $F_i = C_i$. Advance knowledge traffic contracts are used and a static floor limit setting was applied.

In Figure 8.7 the results are shown for the dynamic floor limit method in which the token exchange depends on the actual bucket contents rather than its bucket limit. In this case the same mean page response times are obtained for $F_i \in \{0\%, 100\%\}$ as for the static floor limit setting. As can be observed from the results, the gain becomes far less sensitive to the floor limit. For small traffic contracts and higher values of F_i, TCP connections time-out. This does not occur in the full sharing case.

The difference between the static and dynamic method for setting the floor limit can be explained by the limited token exchange for static systems that have a low token rate in comparison to the bucket limit. This effect is most pronounced for the large bucket limits of 120 and 60000 seconds of token rate that we have simulated. In these cases, even a floor limit of 10% may form a large barrier for token exchange.



(a) $C_1 = C_2 = 10$s. (b) $C_1 = C_2 = 30$s.

Figure 8.7: Improvements in mean page response times for a bucket limit of 10 and 30 seconds. Comparing the policing solution for $F_i \neq 100\%$ of C_i with a system where $F_i = C_i$. Advance knowledge traffic contracts are used and a dynamic floor limit setting was applied.

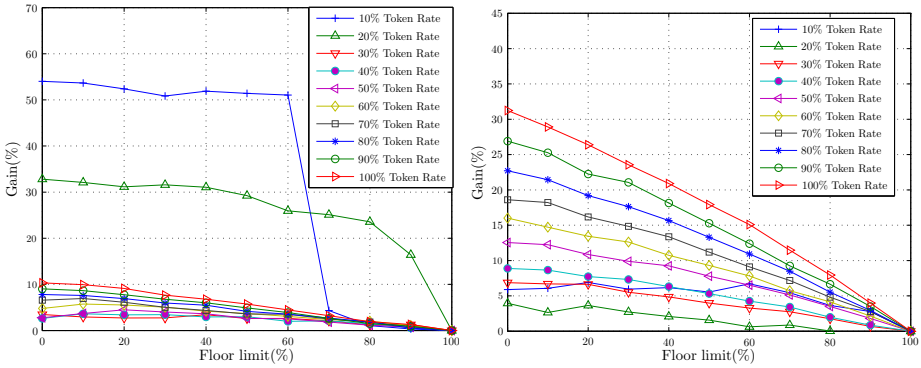
8.3.4 Results for symmetric contracts

When distributing the tokens equally among the up-and downstream directions (referred to as the symmetric contract), less TCP time-outs occur. As a result, response times are shorter and web pages can be retrieved for even small token rates. Comparing Figures 8.6a-8.7b with Figures 8.8a-8.9b, leads to more observations. First, for a small token rate and a small bucket size the advanced knowledge contract yields higher improvements than the symmetric contract. This is caused by the high page response times for the traditional policing for advanced knowledge contracts and small token rates.

Second, when contracts for both directions are equal, high token rates yield the highest relative gains (whereas for advanced knowledge contract, the lower rates yield more gain). This can be observed by comparing the order of the curves for the symmetric contract with the order of the curves for the advanced knowledge contract. Finally, comparing Figures 8.8b and 8.9a show that computing floor limits dynamically results in gains that are less sensitive of the floor limit.

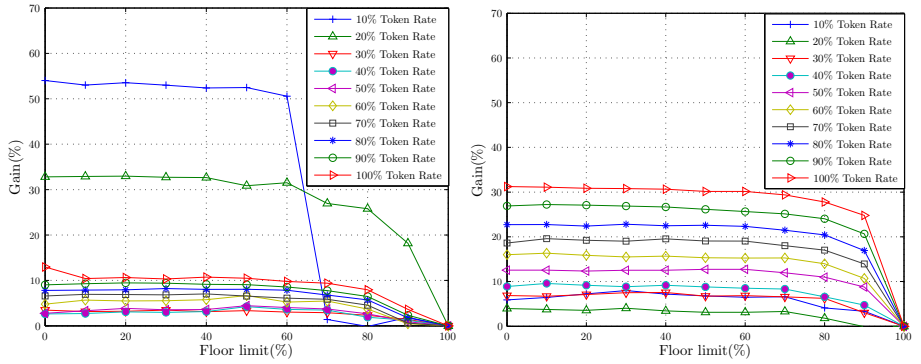
As the bucket limit is further increased, the policing solution shows a stable gain for the symmetric contract when the floor limits are determined dynamically, as a function of the actual token level. For the advanced knowledge traffic contract and the case that the floor limits are computed as function of the bucket limit, the policing mechanism is capable of supporting longer traffic bursts, and thus loose fewer tokens, which results in the traditional approach of policing traffic (and hence lower gains). Sharing the same contract among 5

or 10 wireless users (not shown) has the same effect; the gain vanishes because active flows can grab the tokens from silent ones and hence the exchange of tokens from the other direction does not occur.



(a) $C_1 = C_2 = 10$s. (b) $C_1 = C_2 = 30$s.

Figure 8.8: Improvements in mean page response times for a bucket limit of 10 and 30 seconds. Comparing the policing solution for $F_i \neq 100\%$ of C_i with a system where $F_i = C_i$. Symmetric traffic contracts are used and a static floor limit setting was applied.



(a) $C_1 = C_2 = 10$s. (b) $C_1 = C_2 = 30$s.

Figure 8.9: Improvements in mean page response times for a bucket limit of 10 and 30 seconds. Comparing the policing solution for $F_i \neq 100\%$ of C_i with a system where $F_i = C_i$. Symmetric traffic contracts are used and a dynamic floor limit setting was applied.

8.4 Conclusions and further research

This chapter shows that regulating a traffic contract in two directions concurrently improves the effective usage of the traffic contract. Furthermore, it reduces TCP connection starvation.

The solution described maintains the commonly used policing mechanism in both directions and allows the flexible exchange of transmission tokens between them. In an attempt to prevent that one direction suffocates as a result of favoring the passing of packets of the other direction, a floor limit F was introduced that marks a boundary to the token level, preventing the other direction from grabbing tokens. No transmission tokens can be taken by the other direction if the token level is below the threshold level F. The numerical evaluation of an analytical model that studies performance behavior at packet level shows that by setting the floor limit F at 20% of the capacity of the bucket limit, a good trade-off is achieved between on one hand optimizing performance and on the other preventing that one direction uses the total traffic contract disproportionally.

A simulation analysis was performed to study performance effects on flow-level and reveals that the mechanism improves in particularly the performance for individual traffic contracts and for small traffic contract tolerances. In general, the positive effect in performance vanishes as the number of traffic flows increases that belong to the same contract. It then converges to the traditional approach for policing traffic. In QoS enabled WLAN environments, the per-flow contract between end-user and WLAN operator is common, limiting the number of simultaneous traffic flows per contract considerably. Exactly this circumstance provides a performance improvement when compared to using the traditional policing mechanism. Furthermore, policing traffic in two directions concurrently makes it possible to distinguish services within a traffic contract. A floor limit can be associated with each traffic class, allowing only high priority packets to pass when the leaky buckets is almost empty. One may even consider going one step further and favor high-priority packets over low-priority packets that travel in the opposite direction.

The applicability of the mechanism that polices in two directions concurrently is not restricted to shared wireless access media. It can also be applied in situations where a network operator wants to offer its different wireless transmission technologies to a single user. Network operators with various technologies installed, gain by offering access trough these technologies simultaneously. This requires a single (up/down) traffic contract and hence a mechanism to police such a contract. The mechanism should then be executed in a network device where the streams to and from the different wireless technologies can be distinguished from one and another. If, however, more than two transmission technologies are offered simultaneously, the mechanism as described and evaluated needs to be extended to multiple dimensions.

Appendix A

Maximum of Exponential Random Variables

For self-containedness of the thesis, in this appendix we formulate a known result giving the expected value of the maximum of an arbitrary number of exponentially distributed random variables. This result is used in Section 4.3.3.

Property 1: *If $X_1, \ldots, X_N$ are i.i.d. exponentially distributed random variables with means $1/\mu_1, \ldots, 1/\mu_N$ respectively, then*

$$E\left[\max\{X_1, \ldots, X_N\}\right] = \sum_{k=1}^{N} (-1)^{k+1} \sum_{(i_1,\ldots,i_k)\in S_k} \frac{1}{\mu_{i_1} + \cdots + \mu_{i_k}}, \qquad \text{(A.1)}$$

where S_k $(k = 1, \ldots, N)$ is defined in (4.16).

Proof: The derivation of Property 1 requires only standard algebraic manipulations and is therefore omitted.

To illustrate Property 1, let us work out (A.1) for the cases $N = 2$, 3 and 4. For $N = 2$, we have $S = \{1, 2\}$, $S_1 = \{1, 2\}$, and $S_2 = \{(1, 2)\}$, so that

$$\mathbb{E}\left[\max\{X_1, X_2\}\right] = \frac{1}{\mu_1} + \frac{1}{\mu_2} - \frac{1}{\mu_1 + \mu_2}. \qquad \text{(A.2)}$$

For $N = 3$, we have $S = \{1, 2, 3\}$, $S_1 = \{1, 2, 3\}$, $S_2 = \{(1, 2), (1, 3), (2, 3)\}$ and $S_3 = \{(1, 2, 3)\}$, so that

$$\begin{aligned} \mathbb{E}\left[\max\{X_1, X_2, X_3\}\right] &= \left(\frac{1}{\mu_1} + \frac{1}{\mu_2} + \frac{1}{\mu_3}\right) \\ &- \left(\frac{1}{\mu_1 + \mu_2} + \frac{1}{\mu_1 + \mu_3} + \frac{1}{\mu_2 + \mu_3}\right) \\ &+ \frac{1}{\mu_1 + \mu_2 + \mu_3}. \end{aligned} \qquad \text{(A.3)}$$

Finally, it is readily verified that for $N = 4$ we have $S = \{1,2,3,4\}$, $S_1 = \{1,2,3,4\}$, $S_2 = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$,
$S_3 = \{(1,2,3),(1,2,4),(1,3,4),(2,3,4)\}$, $S_4 = \{(1,2,3,4))\}$, so that

$$\begin{aligned}
&\mathbb{E}\left[\max\{X_1,X_2,X_3,X_4\}\right] = \left(\frac{1}{\mu_1}+\frac{1}{\mu_2}+\frac{1}{\mu_3}+\frac{1}{\mu_4}\right) \qquad \text{(A.4)}\\
-\quad &\left(\frac{1}{\mu_1+\mu_2}+\frac{1}{\mu_1+\mu_3}+\frac{1}{\mu_1+\mu_4}+\frac{1}{\mu_2+\mu_3}+\frac{1}{\mu_2+\mu_4}+\frac{1}{\mu_3+\mu_4}\right)\\
+\quad &\left(\frac{1}{\mu_1+\mu_2+\mu_3}+\frac{1}{\mu_1+\mu_2+\mu_4}+\frac{1}{\mu_1+\mu_3+\mu_4}+\frac{1}{\mu_2+\mu_3+\mu_4}\right)\\
-\quad &\frac{1}{\mu_1+\mu_2+\mu_3+\mu_4}.
\end{aligned}$$

Appendix B

Assignment and splitting ratios of concurrent access policies

This appendix shows the mean traffic assignment or splitting ratios of foreground traffic that is distributed in the presence of background traffic in a simulated CA queueing network with $N = 2$ PS-queues, each with unit capacity. Several CA strategies have been simulated: the static job-split strategy outlined in Chapters 3 and 4, the dynamic job-assignment strategy presented in Chapter 5, the dynamic job-split strategy presented in Chapter 6. As a benchmark that is well-known in the literature, simulation results on the Join-the-Shortest-Queue (JSQ) approach are included. The mean values are obtained from simulation runs of 10^8 foreground observations using Poisson arrivals and exponential jobs-size distributions with unit mean for all traffic streams. The load of the foreground traffic ρ_0 was varied from light ($\rho_0 = 0.1$) to mild ($\rho_0 = 0.5$) and moderate ($\rho_0 = 0.9$) and the background loads ρ_1 and ρ_2 were varied as $0.1, 0.2, \ldots, 0.9$. Due to the symmetry of the concurrent access network, the values are omitted for $\rho_1 > \rho_2$.

B.1 Light foreground traffic load ($\rho_0 = 0.1$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.50	0.50	0.55	0.64	0.70	0.78	0.85	0.92	0.97
0.2		0.50	0.51	0.59	0.67	0.75	0.83	0.91	0.97
0.3			0.50	0.54	0.63	0.71	0.80	0.89	0.96
0.4				0.50	0.56	0.66	0.76	0.87	0.95
0.5					0.50	0.59	0.70	0.83	0.94
0.6						0.50	0.62	0.77	0.91
0.7							0.50	0.66	0.86
0.8								0.50	0.74
0.9									0.50

Table B.1: Split ratio of jobs to PS-queue 1 for the static split policy with light foreground traffic ($\rho_0 = 0.1$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.495	0.842	0.854	0.867	0.883	0.903	0.939	0.959	0.996
0.2		0.506	0.785	0.806	0.829	0.856	0.890	0.936	0.990
0.3			0.488	0.738	0.768	0.803	0.850	0.909	0.958
0.4				0.489	0.700	0.741	0.791	0.859	0.941
0.5					0.479	0.671	0.727	0.802	0.906
0.6						0.453	0.651	0.734	0.859
0.7							0.497	0.648	0.797
0.8								0.515	0.696
0.9									0.500

Table B.2: Assignment ratio of jobs to PS-queue 1 for the dynamic assignment policy with light foreground traffic ($\rho_0 = 0.1$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.521	0.544	0.571	0.601	0.638	0.682	0.740	0.823
0.2		0.500	0.523	0.550	0.581	0.618	0.664	0.724	0.811
0.3			0.500	0.527	0.558	0.596	0.643	0.705	0.796
0.4				0.500	0.532	0.570	0.618	0.682	0.778
0.5					0.500	0.538	0.587	0.654	0.756
0.6						0.500	0.550	0.618	0.727
0.7							0.500	0.570	0.686
0.8								0.500	0.623
0.9									0.500

Table B.3: Split ratio of jobs to PS-queue 1 for the dynamic split policy with light foreground traffic ($\rho_0 = 0.1$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.548	0.597	0.648	0.700	0.755	0.813	0.872	0.935
0.2		0.500	0.550	0.603	0.659	0.718	0.781	0.849	0.922
0.3			0.500	0.554	0.612	0.676	0.745	0.821	0.906
0.4				0.500	0.560	0.626	0.701	0.787	0.885
0.5					0.500	0.569	0.649	0.743	0.858
0.6						0.500	0.583	0.687	0.820
0.7							0.500	0.610	0.765
0.8								0.500	0.674
0.9									0.500

Table B.4: Assignment ratio of jobs to PS-queue 1 for JSQ policy with light foreground traffic ($\rho_0 = 0.1$).

Mild foreground traffic load ($\rho_0 = 0.5$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.50	0.51	0.57	0.65	0.68	0.74	0.81	0.88	0.95
0.2		0.50	0.53	0.59	0.65	0.71	0.78	0.86	0.94
0.3			0.50	0.54	0.60	0.67	0.74	0.83	0.92
0.4				0.50	0.56	0.62	0.70	0.79	0.89
0.5					0.50	0.57	0.65	0.74	0.85
0.6						0.50	0.58	0.68	
0.7							0.50		

Table B.5: Split ratio of jobs to PS-queue 1 for the static split policy with mild foreground traffic ($\rho_0 = 0.5$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.626	0.652	0.671	0.693	0.720	0.754	0.802	0.874	0.949
0.2		0.488	0.621	0.648	0.678	0.716	0.765	0.844	0.928
0.3			0.474	0.597	0.632	0.673	0.724	0.801	0.907
0.4				0.479	0.579	0.625	0.680	0.763	0.879
0.5					0.434	0.567	0.629	0.729	0.843
0.6						0.476	0.569	0.676	
0.7							0.500		

Table B.6: Assignment ratio of jobs to PS-queue 1 for the dynamic assignment policy with mild foreground traffic ($\rho_0 = 0.5$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.522	0.546	0.575	0.608	0.647	0.696	0.760	0.851
0.2		0.500	0.525	0.553	0.587	0.628	0.678	0.746	0.842
0.3			0.500	0.529	0.563	0.605	0.658	0.729	0.833
0.4				0.500	0.534	0.577	0.632	0.708	0.822
0.5					0.500	0.543	0.601	0.683	0.810
0.6						0.500	0.559	0.649	
0.7							0.500		

Table B.7: Split ratio of jobs to PS-queue 1 for the dynamic split policy with mild foreground traffic ($\rho_0 = 0.5$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.539	0.580	0.624	0.671	0.723	0.781	0.845	0.917
0.2		0.500	0.542	0.587	0.636	0.690	0.752	0.822	0.903
0.3			0.500	0.546	0.596	0.653	0.718	0.794	0.886
0.4				0.500	0.551	0.610	0.679	0.761	0.864
0.5					0.500	0.560	0.632	0.721	0.836
0.6						0.500	0.574	0.669	
0.7							0.500		

Table B.8: Assignment ratio of jobs to PS-queue 1 for the JSQ policy with mild foreground traffic ($\rho_0 = 0.5$).

Moderate foreground traffic load ($\rho_0 = 0.9$)

For a moderate foreground traffic load of $\rho_0 = 0.9$ the background traffic load should be less than 0.1 when using the static traffic assignment policy. We have therefore not included any results for this policy here.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.5	0.53	0.57	0.61	0.66	0.71	0.77	0.84	0.91
0.2		0.5	0.54	0.58	0.63	0.68	0.74	0.81	
0.3			0.5	0.54	0.59	0.65	0.71		
0.4				0.5	0.55	0.6			
0.5					0.5				

Table B.9: Split ratio of jobs to PS-queue 1 for the static split policy with moderate foreground traffic ($\rho_0 = 0.9$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.538	0.561	0.586	0.614	0.649	0.697	0.760	0.839	0.907
0.2		0.477	0.548	0.578	0.619	0.668	0.741	0.810	
0.3			0.454	0.539	0.581	0.638	0.711		
0.4				0.493	0.558	0.609			
0.5					0.501				

Table B.10: Assignment ratio of jobs to PS-queue 1 for the dynamic assignment policy with moderate foreground traffic ($\rho_0 = 0.9$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.523	0.550	0.581	0.617	0.662	0.719	0.792	0.889
0.2		0.500	0.527	0.558	0.596	0.643	0.704	0.784	
0.3			0.500	0.532	0.571	0.621	0.686		
0.4				0.500	0.540	0.592			
0.5					0.500				

Table B.11: Split ratio of jobs to PS-queue 1 for the dynamic split policy with moderate foreground traffic ($\rho_0 = 0.9$).

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.500	0.534	0.570	0.610	0.654	0.703	0.759	0.824	0.902
0.2		0.500	0.537	0.577	0.622	0.673	0.732	0.803	
0.3			0.500	0.541	0.587	0.640	0.702		
0.4				0.500	0.547	0.601			
0.5					0.500				

Table B.12: Assignment ratio of jobs to PS-queue 1 for the JSQ policy with moderate foreground traffic ($\rho_0 = 0.9$).

Appendix C

Foreground traffic performance of concurrent access policies

This appendix shows the mean sojourn times (in seconds) of foreground traffic that is distributed in the presence of background traffic in a simulated CA queueing network with $N = 2$ PS-queues, each with unit capacity. Several CA strategies have been simulated: the static job-split strategy outlined in Chapters 3 and 4, the dynamic job-assignment strategy presented in Chapter 5, the dynamic job-split strategy presented in Chapter 6. As a benchmark that is well-known in the literature, simulation results on the Join-the-Shortest-Queue (JSQ) approach are included. The mean values are obtained from simulation runs of 10^8 foreground observations using Poisson arrivals and exponential jobs-size distributions with unit mean for all traffic streams. The load of the foreground traffic ρ_0 was varied from light ($\rho_0 = 0.1$) to mild ($\rho_0 = 0.5$) and moderate ($\rho_0 = 0.9$) and the background loads ρ_1 and ρ_2 were varied as $0.1, 0.2, \ldots, 0.9$. Due to the symmetry of the concurrent access network, the values are omitted for $\rho_1 > \rho_2$.

C.1 Light foreground traffic load ($\rho_0 = 0.1$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.643	0.714	0.801	0.879	0.955	1.031	1.102	1.164	1.216
0.2		0.778	0.865	0.961	1.051	1.146	1.237	1.319	1.386
0.3			0.944	1.054	1.169	1.288	1.406	1.517	1.609
0.4				1.163	1.308	1.462	1.625	1.783	1.916
0.5					1.473	1.682	1.912	2.152	2.366
0.6						1.953	2.296	2.688	3.081
0.7							2.812	3.503	4.350
0.8								4.808	7.006
0.9									14.764

Table C.1: Mean sojourn times of light foreground traffic ($\rho_0 = 0.1$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.073	1.093	1.117	1.141	1.166	1.190	1.210	1.228	1.245
0.2		1.161	1.196	1.234	1.273	1.313	1.351	1.383	1.415
0.3			1.282	1.333	1.392	1.454	1.516	1.575	1.628
0.4				1.442	1.523	1.616	1.715	1.816	1.915
0.5					1.674	1.803	1.958	2.128	2.315
0.6						2.031	2.259	2.548	2.904
0.7							2.652	3.145	3.868
0.8								4.072	5.723
0.9									11.028

Table C.2: Mean sojourn times of light foreground traffic ($\rho_0 = 0.1$) using the dynamic assignment policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.573	0.600	0.630	0.664	0.704	0.751	0.809	0.885	0.997
0.2		0.630	0.665	0.705	0.751	0.806	0.875	0.967	1.104
0.3			0.705	0.751	0.805	0.871	0.954	1.066	1.239
0.4				0.805	0.870	0.949	1.051	1.191	1.413
0.5					0.947	1.045	1.172	1.352	1.648
0.6						1.167	1.331	1.572	1.988
0.7							1.553	1.895	2.530
0.8								2.430	3.560
0.9									6.547

Table C.3: Mean sojourn times of light foreground traffic ($\rho_0 = 0.1$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.074	1.113	1.151	1.187	1.220	1.248	1.269	1.280	1.276
0.2		1.164	1.216	1.267	1.316	1.362	1.402	1.431	1.444
0.3			1.283	1.353	1.423	1.493	1.560	1.617	1.658
0.4				1.445	1.543	1.645	1.750	1.853	1.943
0.5					1.675	1.821	1.983	2.159	2.339
0.6						2.029	2.276	2.572	2.923
0.7							2.653	3.161	3.880
0.8								4.075	5.734
0.9									11.028

Table C.4: Mean sojourn times of light foreground traffic ($\rho_0 = 0.1$) using the JSQ policy.

C.2 Mild foreground traffic load ($\rho_0 = 0.5$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.855	0.977	1.112	1.253	1.411	1.596	1.806	2.042	2.294
0.2		1.085	1.240	1.415	1.616	1.860	2.155	2.513	2.943
0.3			1.399	1.617	1.882	2.214	2.650	3.235	4.049
0.4				1.878	2.235	2.713	3.397	4.447	6.303
0.5					2.726	3.461	4.641	6.896	13.192
0.6						4.691	7.119	14.211	
0.7							14.487		

Table C.5: Mean sojourn times of mild foreground traffic ($\rho_0 = 0.5$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.235	1.287	1.363	1.453	1.560	1.687	1.830	1.998	2.243
0.2		1.386	1.482	1.598	1.738	1.910	2.124	2.387	2.790
0.3			1.632	1.771	1.958	2.200	2.521	2.955	3.676
0.4				1.994	2.245	2.593	3.094	3.864	5.343
0.5					2.654	3.172	4.022	5.592	9.754
0.6						4.111	5.841	10.393	
0.7							11.082		

Table C.6: Mean sojourn times of mild foreground traffic ($\rho_0 = 0.5$) using the dynamic assignment policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	0.744	0.792	0.847	0.911	0.990	1.088	1.217	1.403	1.713
0.2		0.848	0.914	0.993	1.090	1.215	1.384	1.636	2.080
0.3			0.994	1.092	1.217	1.381	1.611	1.973	2.663
0.4				1.218	1.381	1.607	1.942	2.510	3.755
0.5					1.607	1.937	2.475	3.530	6.669
0.6						2.473	3.511	6.412	
0.7							6.506		

Table C.7: Mean sojourn times of mild foreground traffic ($\rho_0 = 0.5$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.224	1.299	1.382	1.476	1.584	1.708	1.855	2.029	2.238
0.2		1.390	1.495	1.616	1.759	1.932	2.146	2.422	2.796
0.3			1.627	1.784	1.975	2.220	2.541	2.997	3.705
0.4				1.990	2.254	2.609	3.119	3.926	5.497
0.5					2.630	3.170	4.047	5.737	10.695
0.6						4.079	5.833	10.967	
0.7							11.039		

Table C.8: Mean sojourn times of mild foreground traffic ($\rho_0 = 0.5$) using the JSQ policy.

C.3 Moderate foreground traffic load ($\rho_0 = 0.9$)

For a moderate foreground traffic load of $\rho_0 = 0.9$ the background traffic load should be less than 0.1 when using the static traffic assignment policy. We have therefore not included any results for this policy here.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.264	1.503	1.783	2.137	2.613	3.311	4.421	6.533	12.429
0.2		1.755	2.120	2.610	3.329	4.502	6.781	13.426	
0.3			2.597	3.328	4.524	6.911	13.882		
0.4				4.526	6.913	14.231			
0.5					14.089				

Table C.9: Mean sojourn times of moderate foreground traffic ($\rho_0 = 0.9$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.592	1.744	1.932	2.185	2.528	2.997	3.744	5.197	8.877
0.2		1.954	2.214	2.572	3.080	3.876	5.330	9.190	
0.3			2.603	3.141	3.995	5.564	9.653		
0.4				4.085	5.651	10.328			
0.5					10.820				

Table C.10: Mean processing time of foreground jobs for the dynamic Bayesian traffic assignment policy to PS-queue 1 in an $N = 2$ concurrent access network with a light foreground load of $\rho_0 = 0.9$. The outcomes are obtained from simulation in which jobs are exponentially distributed.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.065	1.170	1.300	1.467	1.694	2.023	2.556	3.606	6.758
0.2		1.304	1.477	1.712	2.051	2.588	3.600	6.331	
0.3			1.719	2.071	2.631	3.687	6.546		
0.4				2.650	3.764	6.901			
0.5					7.060				

Table C.11: Mean sojourn times of moderate foreground traffic ($\rho_0 = 0.9$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.568	1.723	1.919	2.175	2.522	3.039	3.880	5.544	10.490
0.2		1.925	2.186	2.545	3.071	3.936	5.633	10.686	
0.3			2.551	3.088	3.961	5.681	10.747		
0.4				3.970	5.709	10.815			
0.5					10.771				

Table C.12: Mean sojourn times of moderate foreground traffic ($\rho_0 = 0.9$) using the JSQ policy.

Appendix D

Background traffic performance of concurrent access policies

This appendix shows the background traffic performance, denoted γ, that is defined by $\gamma = \frac{\mathbb{E}[S_1]\rho_1 + \mathbb{E}[S_2]\rho_2}{\rho_1+\rho_2}$ in the presence of foreground traffic that is distributed in a simulated CA queueing network with $N = 2$ PS-queues, each with unit capacity. Several CA strategies have been simulated: the static job-split strategy outlined in Chapters 3 and 4, the dynamic job-assignment strategy presented in Chapter 5, the dynamic job-split strategy presented in Chapter 6. As a benchmark that is well-known in the literature, simulation results on the Join-the-Shortest-Queue (JSQ) approach are included. The mean values are obtained from simulation runs of 10^8 foreground observations using Poisson arrivals and exponential jobs-size distributions with unit mean for all traffic streams. The load of the foreground traffic ρ_0 was varied from light ($\rho_0 = 0.1$) to mild ($\rho_0 = 0.5$) and moderate ($\rho_0 = 0.9$) and the background loads ρ_1 and ρ_2 were varied as $0.1, 0.2, \ldots, 0.9$. Due to the symmetry of the concurrent access network, the values are omitted for $\rho_1 > \rho_2$.

D.1 Light foreground traffic load ($\rho_0 = 0.1$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.176	1.281	1.441	1.658	1.974	2.441	3.224	4.767	9.403
0.2		1.333	1.456	1.642	1.919	2.345	3.058	4.471	8.693
0.3			1.538	1.695	1.939	2.327	2.984	4.294	8.226
0.4				1.818	2.035	2.388	3.000	4.215	7.897
0.5					2.222	2.550	3.129	4.285	7.719
0.6						2.857	3.421	4.555	7.888
0.7							4.000	5.207	8.585
0.8								6.667	10.889
0.9									20.000

Table D.1: Background traffic performance with light foreground traffic ($\rho_0 = 0.1$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.147	1.236	1.381	1.588	1.886	2.337	3.086	4.599	9.140
0.2		1.297	1.404	1.577	1.838	2.244	2.930	4.303	8.452
0.3			1.490	1.631	1.859	2.224	2.850	4.114	7.950
0.4				1.752	1.948	2.277	2.856	4.033	7.577
0.5					2.125	2.421	2.959	4.069	7.386
0.6						2.698	3.203	4.265	7.431
0.7							3.698	4.750	7.877
0.8								5.891	9.346
0.9									15.151

Table D.2: Background traffic performance with light foreground traffic ($\rho_0 = 0.1$) using the dynamic assignment policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.160	1.260	1.416	1.637	1.956	2.441	3.259	4.912	9.913
0.2		1.312	1.429	1.614	1.894	2.332	3.080	4.605	9.245
0.3			1.509	1.660	1.904	2.298	2.985	4.400	8.738
0.4				1.777	1.986	2.340	2.973	4.296	8.393
0.5					2.160	2.476	3.062	4.308	8.231
0.6						2.754	3.296	4.485	8.285
0.7							3.798	4.953	8.763
0.8								6.129	10.231
0.9									16.347

Table D.3: Background traffic performance with light foreground traffic ($\rho_0 = 0.1$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.146	1.242	1.392	1.602	1.903	2.358	3.118	4.638	9.191
0.2		1.295	1.408	1.586	1.851	2.261	2.953	4.346	8.519
0.3			1.489	1.635	1.867	2.237	2.872	4.156	8.014
0.4				1.750	1.952	2.286	2.872	4.065	7.655
0.5					2.123	2.425	2.969	4.089	7.460
0.6						2.697	3.208	4.280	7.501
0.7							3.695	4.757	7.936
0.8								5.885	9.373
0.9									15.148

Table D.4: Background traffic performance with light foreground traffic ($\rho_0 = 0.1$) using the JSQ policy.

D.2 Mild foreground traffic load ($\rho_0 = 0.5$)

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.538	1.718	1.953	2.290	2.749	3.446	4.520	6.591	12.273
0.2		1.819	2.038	2.348	2.799	3.503	4.641	6.691	12.248
0.3			2.222	2.541	3.021	3.750	5.026	7.280	13.555
0.4				2.858	3.373	4.237	5.700	8.639	17.369
0.5					4.002	5.063	7.046	11.757	30.607
0.6						6.667	10.182	21.369	
0.7							19.984		

Table D.5: Background traffic performance with mild foreground traffic ($\rho_0 = 0.5$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.374	1.482	1.669	1.934	2.311	2.875	3.729	5.338	9.634
0.2		1.578	1.740	1.983	2.345	2.894	3.781	5.373	9.400
0.3			1.878	2.116	2.471	3.037	3.989	5.737	9.679
0.4				2.334	2.715	3.326	4.409	6.526	11.269
0.5					3.104	3.846	5.227	7.930	17.691
0.6						4.759	6.957	12.968	
0.7							11.878		

Table D.6: Background traffic performance with mild foreground traffic ($\rho_0 = 0.5$) using the dynamic assignment policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.434	1.580	1.804	2.125	2.596	3.336	4.626	7.358	16.240
0.2		1.675	1.860	2.150	2.594	3.306	4.575	7.322	16.455
0.3			2.014	2.278	2.705	3.412	4.707	7.591	17.585
0.4				2.528	2.954	3.691	5.086	8.334	20.587
0.5					3.410	4.239	5.908	10.162	29.946
0.6						5.322	7.781	15.970	
0.7							13.756		

Table D.7: Background traffic performance with mild foreground traffic ($\rho_0 = 0.5$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.342	1.467	1.649	1.900	2.251	2.771	3.614	5.255	9.950
0.2		1.558	1.718	1.951	2.289	2.798	3.625	5.206	9.672
0.3			1.860	2.086	2.421	2.939	3.787	5.413	9.847
0.4				2.311	2.667	3.229	4.178	6.011	10.996
0.5					3.075	3.747	4.961	7.519	15.537
0.6						4.697	6.656	12.426	
0.7							11.851		

Table D.8: Background traffic performance with mild foreground traffic ($\rho_0 = 0.5$) using the JSQ policy.

D.3 Moderate foreground traffic load ($\rho_0 = 0.9$)

For a moderate foreground traffic load of $\rho_0 = 0.9$ the background traffic load should be less than 0.1 when using the static traffic assignment policy. We have therefore not included any results for this policy here.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	2.222	2.556	3.042	3.781	4.840	6.714	10.033	16.649	48.409
0.2		2.857	3.372	4.204	5.504	8.030	13.437	30.393	
0.3			4.002	5.078	6.998	10.756	22.800		
0.4				6.672	10.076	21.764			
0.5					19.985				

Table D.9: Background traffic performance with moderate foreground traffic ($\rho_0 = 0.9$) using the static split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.745	2.032	2.391	2.917	3.697	4.841	6.995	9.342	24.560
0.2		2.171	2.556	3.150	4.060	5.654	7.991	16.110	
0.3			2.867	3.597	4.864	7.311	13.331		
0.4				4.399	6.188	12.008			
0.5					11.058				

Table D.10: Background traffic performance with moderate foreground traffic ($\rho_0 = 0.9$) using the dynamic assignment policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.970	2.239	2.651	3.269	4.238	5.909	9.248	18.063	61.726
0.2		2.472	2.884	3.548	4.657	6.707	11.309	26.905	
0.3			3.339	4.133	5.581	8.662	18.090		
0.4				5.257	7.662	14.995			
0.5					14.176				

Table D.11: Background traffic performance with moderate foreground traffic ($\rho_0 = 0.9$) using the dynamic split policy.

ρ_1 \ ρ_2	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	1.709	1.911	2.201	2.605	3.183	4.075	5.593	8.694	18.095
0.2		2.111	2.421	2.883	3.592	4.771	7.051	13.370	
0.3			2.789	3.385	4.383	6.347	11.877		
0.4				4.268	6.064	11.306			
0.5					11.081				

Table D.12: Background traffic performance with moderate foreground traffic ($\rho_0 = 0.9$) using the JSQ policy.

Publications of the author

Scientific publications

1. G.J. Hoekstra, R.D. van der Mei and S. Bhulai. Optimal job splitting in parallel processor sharing queues. To appear in Stochastic Models, 28(1), 2012.

2. G.J. Hoekstra, R.D. van der Mei and J.W. Bosman. On comparing the performance of dynamic multi-network optimizations. In Proceedings IEEE GlobeCom, Miami, U.S.A., 2010.

3. G.J. Hoekstra and R.D. van der Mei. Effective load for flow-level performance modelling of file transfers in wireless LANs. Computer Communications, 33(16):1972-1981, 2010.

4. F.J.M. Panken and G.J. Hoekstra. Multi-service traffic profiles to realise and maintain QoS guarantees in wireless LANs. Computer Communications, 32(6):1022-1033, 2009.

5. S. Bhulai, G.J. Hoekstra and R.D. van der Mei. Optimal concurrent access strategies in mobile communication networks. In Proceedings 22^{nd} International Teletraffic Congress, Amsterdam, The Netherlands, 2010.

6. S. Bhulai, G.J. Hoekstra, J.W. Bosman, and R.D. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A bayesian approach. Performance Evaluation, 69(1):4152, 2012.

7. G.J. Hoekstra, R.D. van der Mei, Y. Nazarathy and A.P. Zwart. Optimal file splitting for wireless networks with concurrent Access. Lecture Notes in Computer Science, 5894:189-203. Springer Verlag, 2009.

8. G.J. Hoekstra and R.D. van der Mei. On the processor sharing of file transfers in wireless LANs. In Proceedings of the 69th IEEE Vehicular Technology Conference, VTC Spring 2009, Barcelona, Spain, 2009.

9. F.J.M. Panken, G.J. Hoekstra, D. Barankanira, J.C. Francis, R. Schwendener, O. Grondalen and M.G. Jaatun. Extending 3G/WiMAX networks and services through residential access capacity. IEEE Communications Magazine, 45(12):62-69, 2007.

10. G.J. Hoekstra and F.J.M. Panken. On the efficient policing of HTTP traffic in WLANs. In Proceedings Third ERCIM Workshop on eMobility, pages 15-29, University of Twente, Enschede, The Netherlands, 2009.

11. G.J. Hoekstra and R.D. van der Mei. On the processor sharing properties of file transfers in a WLAN testbed. In Proceedings Internet-2010, pages 36-41, Valencia, Spain, 2010.

12. G.J. Hoekstra et al. Management of the WDM network layer. Chapter in book Deploying and Managing IP over WDM Networks, Artech House Publishers, 2003.

13. G.J. Hoekstra and F.J.M. Panken. Increasing throughput of data applications on heterogeneous wireless access networks. In Proceedings 12^{th} IEEE Symposium on Communication and Vehicular Technology in the Benelux, 2005.

14. G.J. Hoekstra, R.D. van der Mei, T. Dziedzicki and J.W. Bosman. Efficient traffic splitting in parallel TCP-based networks: Modelling and experimental evaluation. Submitted for publication.

15. F.J.M. Panken and G.J. Hoekstra. On regulating traffic in shared wireless access media. Submitted for publication.

16. F.J.M. Panken, P.E. Bryhni, H. Engelstad, L. Hansson, G.J. Hoekstra, M.G. Jaatun and T.J. Johannessen. Architecture for sharing residential access with roaming WLAN users. Telektronikk, 3:48-59, 2006.

17. F.J.M. Panken, G.J. Hoekstra and T. van der Gaast. Resource allocation and guarantees for real-time applications in WLANs. Telektronikk, 3:125-134, 2006.

18. G.J. Hoekstra, et al. Quality of service solution for open wireless access networks. In Proceedings 14^{th} IST Mobile & Wireless Communications Summit, Dresden, Germany, 2005.

19. G.J. Hoekstra et al. An integrated network management solution for multi-technology domain networks. Bell Labs Technical Journal, 7(1):115-120, 2002.

20. W.A. Romijn, G.J. Hoekstra, J. Serrat, and E. Grampin. Enhancing network management by applying policy management principles. Bell Labs Technical Journal, 8(1):151-156, 2003.

21. J. van Bemmel, H. Teunissen, and G.J. Hoekstra. Security aspects of 4G services. In Proceedings 9^{th} WWRF meeting, Zurich, Switzerland, 2003.

22. D. Miliche, M. de Graaf, G.J. Hoekstra, M. Jongerden and B. Haverkort. A first experimental investigation of the practical efficiency of battery scheduling. In Proceedings Workshop on Ultra-Low Power Sensor Networks (WUPS), pages 241-246. VDE Verlag GmbH, 2010.

23. J. Serrat, E. Grampin, L. Raptis, F. Karayannis, K. Vaxevanakis, D. Chronis, H. Katopodis, G.J. Hoekstra, W. Romijn, A. Galis and E. Kozlovski. Functional evaluation of an integrated IP over WDM management solution. In Proceedings IFIP/IEEE Eighth International Symposium on Integrated Network Management, pages 679-692, 2003.

Patent publications

1. G.J. Hoekstra. Method and device for finding the shortest non-looping paths. Patent, November 2007. EP2063582A1.

2. G.J. Hoekstra and H.B. Meeuwissen. Location estimation in end-user devices using public radio signals. Patent, May 2007. Application No. US 2007/747422 A.

3. T. van der Gaast and G.J. Hoekstra. Method for distributing identical data and different data to mobile units. Patent, May 2007. Application No. US 2007/745760 A.

4. G.J. Hoekstra and F.J.M. Panken. Method of predicting transmission speed adaptations. Patent, August 2006. Application No. US 2006/463389 A.

5. T. van der Gaast and G.J. Hoekstra. Method of distributing identical data to mobile units. Patent, July 2006. Application No. US 2006/461054 A.

6. G.J. Hoekstra and F.J.M. Panken Maintaining quality of service for wireless communications. Patent, June 2006. Application No. US 2006/450546 A.

7. G.J. Hoekstra. Translating network addresses for multiple network interfaces. Patent, November 2005. Application No. US 2005/265006 A.

8. G.J. Hoekstra and H.B. Meeuwissen. Distributing information over parallel network interfaces. Patent, November 2005. Application No. US 2005/264855 A.

9. T. van der Gaast, G.J. Hoekstra, F.J.M. Panken and J. van Bemmel. Method for assigning uplink and/or downlink capacities based on available capacity. Patent, August 2005. Application No. US 2005/204194 A.

10. G.J. Hoekstra and F.J.M. Panken. Method for distributing transport sessions over multiple network interfaces. Patent, February 2005. Application No. US 2006/450546 A.

Samenvatting

De efficiëntie waarmee draadloze communicatie systemen gebruik maken van het frequentie spectrum is over de afgelopen jaren aanzienlijk verbeterd. Hedendaagse coderingstechnieken hebben de theoretische (Shannon) wet zeer dicht kunnen benaderen. Door de toepassing van geavanceerde signaalbewerking in hoogwaardige technologieën heeft dit geresulteerd in de zeer hoge snelheden die door moderne communicatie systemen gerealiseerd kunnen worden. Op deze onderdelen van draadloze communicatiesystemen is naar verwachting voor vergelijkbare omstandigheden slechts door zeer complexe technieken een beperkte verhoging te behalen zonder te afstand te moeten reduceren waarover gecommuniceerd wordt. Dit is een fundamentele limiet op een verdere verhoging van de snelheid van draadloze systemen. Er bestaan veel soorten draadloze netwerken die veelal tegelijk te beschikbaar zijn omdat deze actief zijn in een ander gedeelte van het frequentiespectrum. Dit spectrum is opgedeeld in veelal gelicenseerde frequentiebanden. Uit metingen is gebleken dat dit spectrum niet overal druk bezet is. In sommige banden is nauwelijks activiteit waar te nemen. Het tegelijk gebruiken van meerdere netwerken ligt voor de hand als methode of toch een hogere snelheid te behalen voor draadloze systemen. De potentiële prestatiewinsten van deze methode zijn zeer groot, mits het netwerkverkeer van de gebruikers op een juiste manier over deze netwerken verdeeld wordt.

In dit proefschrift worden methoden bestudeerd voor het optimaliseren van verkeersstromen in draadloze communicatienetwerken. Een aantal van deze methoden is gericht op het verbeteren van de dienstverlening aan gebruikers door de door hen gegenereerde digitale verkeersstromen over *meerdere simultaan beschikbare netwerken* te verdelen en tegelijkertijd op het realiseren van een betere benutting van de beschikbare netwerkcapaciteit. Het gelijktijdig gebruiken van meerdere, parallelle communicatienetwerken wordt in dit proefschift *Concurrent Access* (CA) genoemd.

Voor toepassing in *afzonderlijke* netwerken zijn methoden bestudeerd die aan de hand van de eisen van aanwezige gebruikers enerzijds en een dynamisch variërende beschikbare netwerk capaciteit anderzijds een contract tussen het netwerk en de afzonderlijke gebruikers vaststellen en afdwingen. Het gemeenschappelijke doel van de ontwikkelde methoden is om de dienstverlening zoals deze door de gebruikers ervaren wordt te verbeteren en tegelijkertijd de benut-

ting van de beschikbare netwerken te verhogen.

Het optimaliseren van verkeersstromen over meerdere netwerken wordt gedaan in drie stappen. De eerste stap is het opstellen en valideren van een prestatiemodel van een communicatienetwerk. Het model vormt een abstractie van één afzonderlijk communicatienetwerk die het gedrag ten aanzien het verwerken van verkeersstromen nauwkeurig beschrijft door toepassing van de juiste parameterisering. In de tweede stap wordt een prestatiemodel bestudeerd waarin het bedrag van meerdere, parallelle communicatienetwerken in verwerkt is. Dit model wordt vervolgens geanalyseerd en geoptimaliseerd. In de derde en laatste stap worden de ontwikkelde methoden geëvalueerd aan de hand van simulaties en experimenten met communicatiesystemen.

Hoofdstuk 1 van dit proefschrift bevat een beschrijving van de achtergrond en de motivatie van het verrichte onderzoek, en geeft een overzicht van de beschikbare literatuur op het gebied van communicatienetwerken en prestatiemodellen.

In hoofdstuk 2 wordt een nieuw concept geïntroduceerd voor het modelleren van verkeersstromen in een communicatienetwerk met elastisch data-verkeer dat wordt afgehandeld door middel van het zogenaamde Transport Control Protocol (TCP). De invloed van de complexe, gecombineerde interactie van meerdere protocollagen in communicatie systemen op de verwerking van bestandsstromen kan als expliciete uitdrukking met één enkele parameter, genaamd de *effectieve bedieningsduur*, worden beschreven. Gebaseerd op deze effectieve bedieningsduur van een aankomstenpatroon van bestandsstromen kan de *effectieve belasting* in een netwerk gedefinieërd worden. Op basis daarvan kan de door de gebruiker ervaren prestatie van het netwerk beschreven worden door middel van een $M/G/1$ Processor Sharing (PS) model. De resultaten van een uitgebreide validatie van dit model door middel van simulaties en experimenten geven aan dat nauwkeurige voorspellingen van de netwerkprestaties verkregen kunnen worden voor een uiteenlopende set van parameters. Dit stelt ons in staat de prestatie van complexe draadloze netwerken te beschrijven door middel van een eenvoudig model dat vervolgens kan worden gebruikt voor het evalueren en optimaliseren van methoden voor het efficient splitsen van verkeersstromen.

In hoofdstuk 3 modelleren we het gebruik van meerdere parallelle communicatienetwerken door middel van een CA-model waarbij elk netwerk door een PS-model wordt gerepresenteerd. In het model worden twee typen verkeer bediend: voorgrondverkeer en achtergrondverkeer. De taken van het voorgrondverkeer worden opgesplitst in fragmenten volgens een vaste splitsregel. Vervolgens worden de fragmenten verwerkt door de parallelle PS-nodes. Zodra alle fragmenten van een taak zijn verwerkt wordt de oorspronkelijke taak gereconstrueerd waarmee de behandeling is beëindigd. Elke PS-node ontvangt een stroom achtergrondverkeer die niet gesplitst wordt. Vanwege het toepassen van een vaste splitsregel op de taken van het voorgrondverkeer wordt deze methode *static job splitting* genoemd. De doelstelling van het statisch splitsen van

taken is om de behandelingsduur van voorgrondtaken te bekorten. Door gebruik te maken van het PS-model met de effectieve bedieningsduur uit hoofdstuk 2 is het model, genaamd het *CA job-split model*, toe te passen op het splitsen van bestandsstromen over meerdere parallelle communicatienetwerken. Hoofdstuk 3 richt zich in het bijzonder op het gedrag van zeer grote taken die volgens een 'zwaartstaartige' verdeling aan het systeem worden aangeboden zodat van asymptotische eigenschappen gebruik gemaakt kan worden. Op basis van een zogenaamde Reduced Load Approximation (RLA) wordt bewezen dat een zeer eenvoudige en intuïtieve splitsregel tot optimale prestaties leidt ten aanzien van de bedieningsduur van zeer grote taken uit het voorgrondverkeer. Resultaten verkregen via uitvoerige simulaties demonsteren dat deze eenvoudige splitsregel ook goed presteert ten aanzien van de gemiddelde behandelingsduur van het voorgrondverkeer.

In hoofdstuk 4 wordt het CA job-split model van hoofdstuk 3 verder geanalyseerd. Het doel is een splitsregel te ontwikkelen en te evalueren voor de *static job splitting* methode zodanig dat de gemiddelde behandelingsduur van het gesplitste voorgrondverkeer geminimaliseerd wordt. Vanwege het splitsen van een aankomststroom van voorgrondtaken raken de parallelle PS-nodes van het CA job-split model doorgaans gecorreleerd, hetgeen een exacte analyse van de resulterende gemiddelde bedieningsduren moeilijk maakt. Daarom wordt in hoofdstuk 4 een benadering voor de optimale splitsregel ontwikkeld die gebaseerd is op een combinatie van twee methoden; de eerste methode berust op de asymptotische eigenschappen van zwaarstaartige bedieningsduurverdelingen en de tweede methode op de eigenschappen van lichtstaartige verdelingen. Uit simulatieresultaten blijkt dat door toepassing van de ontwikkelde splitsregel de gemiddelde bedieningsduur van het voorgrondverkeer de geschatte prestaties van een optimale splitsregel zeer dicht benaderen.

In hoofdstuk 5 wordt een CA job-assignment model geanalyseerd, waarbij taken van het voorgrond verkeer, in de aanwezigheid van achtergrond verkeer, dynamisch kunnen worden toegewezen aan een van de PS-nodes van het systeem. Deze methode om voorgrondverkeer aan één van de PS-nodes van het CA job-assignment model toe te wijzen wordt in de context van dit proefschrift aangeduid als *dynamic job assignment*. Het doel van dit hoofdstuk is om toewijzingsregels te ontwikkelen en evalueren voor de dynamic job assignment methode zodanig dat de gemiddelde behandelingsduur van het voorgrondverkeer geminimaliseerd wordt. Voor dit toewijzingsprobleem is een zogenaamd Markov-beslismodel opgesteld en opgelost dat op basis van de volledige toestandsruimte van het systeem in staat is om de taken dusdanig toe te wijzen aan de juiste PS-node dat dit tot optimale prestaties leidt. Echter, onder realistische omstandigheden is vaak slechts een deel van de toestandsruimte bekend. Daarom is in dit hoofdstuk een zogenaamd Bayesiaans algoritme ontwikkeld dat op basis van partiële informatie over de toestand van het systeem de ontbrekende toestandsvariabelen kan schatten. Zodoende kan gebruik gemaakt worden van het Markov-beslismodel die berust op de volledige toestandsruimte van het sys-

teem. Op basis van simulaties wordt aangetoond dat dynamische toewijzing op basis van het Bayesiaanse algoritme in zowel een CA job-assignment model als in een vergelijkbare draadloze netwerkopgeving de "ideale" prestaties van een Markov-model met volledige toestandsruimte zeer dicht benaderen.

In hoofdstuk 6 wordt een CA job-split model geanalyseerd, waarbij de taken van het voorgrondverkeer worden gesplitst in fragmenten die tijdens de bediening in grootte kunnen variëren. Deze fragmenten zullen zich dusdanig aanpassen dat de individuele behandelingsduur van alle fragmenten gelijk is. Vanwege het toepassen van een dynamische splitsregel op iedere taak van het voorgrondverkeer wordt deze methode *dynamic job splitting* genoemd. Bij toepassing van deze methode in het CA job-split model wordt aangenomen dat de splitsregel gebaseerd is op de volledige toestandsruimte van het model en dat taken met oneindig fijne granulariteit kunnen worden gesplitst, als poging om "optimale" prestaties van het voorgrondverkeer bereiken. Naast een model wordt een praktische realisatie van de *dynamic job splitting* methode gepresenteerd. Door gebruik te maken een combinatie van zowel het model uit hoofdstuk 2, voor de effectieve bedieningsduur van een bestandsstroom in één communicatienetwerk, en het prestatie model voor *dynamic job splitting* kunnen de "optimale" prestaties in een realistisch communicatienetwerk bepaald worden. De uitkomsten van dit model worden vervolgens vergeleken met de uitkomsten van de experimenten om de efficiëntie van de praktische oplossing te bepalen die TCP-verkeersstromen over meerdere draadloze netwerken optimaliseert. Uit de uitkomsten van deze experimenten blijkt dat de ontwikkelde *dynamic job splitting* methode voor draadloze netwerken de analytisch bepaalde "optimale" prestaties zeer dicht benadert en daarmee zeer efficiënt verkeersstromen over meerdere netwerken kan verdelen in de aanwezigheid van achtergrondverkeer.

In hoofdstuk 7 worden elementen van het model uit hoofdstuk 2 gebruikt als basis om een oplossing te ontwikkelen die de verwachtingen van gebruikers in een draadloos netwerk kan relateren aan de benodigde netwerkcapaciteit van deze gebruikers. In dit hoofdstuk wordt duidelijk gemaakt dat een capaciteit uitgedrukt in bits per seconde onvoldoende inzicht geeft in de gebruikersbehoefte ten aanzien van bepaalde toepassingen of de voortvloeiende netwerkbelasting van deze toepassingen. De oorzaak hiervan is de verschillende manier waarop netwerken pakketten versturen. In sommige netwerken kan de capaciteit zelfs sterk fluctueren door adaptatie mechanismen. De beslaglegging op de capaciteit kan door deze oorzaken uiteenlopen. De voorgestelde oplossing hiervoor in dit hoofdstuk is het definiëren van een zogenaamd *multi-service traffic profile* dat wél inzicht geeft in de behoefte van toepassingen en de resulterende belasting ervan in een netwerk. Het multi-service traffic profile wordt gebruikt om een verkeerscontract, zogenaamde *QoS budget*, te bepalen waar het draadloze communicatiesysteem van een gebruiker zich aan dient te houden volgens de methode die beschreven is in [47]. Op deze manier kan dichtbij de gebruiker beslist worden hoe binnen het QoS budget gecommuniceerd kan worden en of een nieuwe toepassing het netwerk mag gebruiken. Het gebruik van de multi-

service traffic profile en het bepalen en toewijzen van een QoS budget kan op een dynamische manier plaatsvinden die past bij een variërende netwerkcapaciteit als gevolg van bijvoorbeeld veranderende kanaalcondities of bewegende gebruikers.

In hoofdstuk 8 wordt een oplossing voorgesteld en geëvalueerd die ontworpen is om een verkeerscontract dat aan een gebruiker is toegewezen af te dwingen in een zogenaamd *shared medium netwerk*. Het verkeerscontract kan hierbij tot stand komen via de methode die voorgesteld is in hoofdstuk 7. Traditioneel worden in een verkeerscontract de stromen van en naar de gebruiker (respectievelijk *upstream* en *downstream* verkeersstromen) onderscheiden en wordt bij het afdwingen van het contract, ofwel *policen*, iedere richting afzonderlijk aan de grenzen uit het contract onderworpen. Voor (veelal traditionele) netwerken die afzonderlijke kanalen gebruiken voor upstream en downstream verkeer, of een andere scheiding aanbrengen, functioneert deze policing methode naar behoren. Bij een shared medium kan dit tot lagere prestaties voor de gebruiker en een inefficiënt gebruik van het netwerk leiden. De voorgestelde policing oplossing maakt gebruik van eigenschappen van shared media om binnen de grenzen van het totale verkeerscontract een andere verhouding tussen upstream en downstream verkeer toe te staan, indien nodig. Op basis van een analytisch model en uitgebreide simulaties kan worden geconcludeerd dat de voorgestelde policing oplossing, ten opzichte van traditionele policing methoden, in een shared medium netwerk de capaciteit efficiënter benut en de dienstverlening aan de gebruiker verbeterd. Tevens wordt aangegeven hoe de oplossing te parameteriseren is.

Bibliography

[1] ANSI/IEEE Standard 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. 1999.

[2] IEEE Standard 802.11a. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High speed physical layer in the 5 GHz band. 1999.

[3] IEEE Standard 802.11b. Higher Speed Physical layer (PHY) extension in the 2.4 GHz band. September 1999.

[4] IEEE Standard 802.11e. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. November 2005.

[5] IEEE Standard 802.11g. Part 11: Wireless LAN Medium Access control (MAC) and Physical Layer (PHY) specifications: Further higher-speed physical layer extension in the 2.4 GHz band. June 2003.

[6] IEEE Standard 802.11n. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer specifications Enhancements for Higher Throughput. October 2009.

[7] IEEE Standard 802.1H. IEEE Standards for Local and Metropolitan Area Networks: Recommended Practice for Media Access Control (MAC) Bridging of Ethernet V2.0 in IEEE 802 Local Area Networks. 1995.

[8] M.H. Ahmed. Call admission control in wireless networks: a comprehensive survey. *IEEE Communications Surveys and Tutorials*, 7(1):49–68, 2005.

[9] S.C. Albright. Structural results for partially observable Markov decision processes. *Operations Research*, 27:1041–1053, 1979.

[10] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. RFC 2414, Internet Engineering Task Force, September 1998.

[11] E. Altman, U. Ayesta, and B. Prabhu. Load balancing in processor sharing systems. In *Proceedings of the* 3^{rd} *International Conference on Performance Evaluation Methodologies and Tools*, ValueTools '08, pages 12:1–12:10, 2008.

[12] F. Baccelli, W.A. Massey, and D.F. Towsley. Acyclic fork-join queuing networks. *Journal of the ACM*, 36(3):615–642, 1989.

[13] A. Banchs, X. Pérez-costa, and D. Qiao. Providing throughput guarantees in IEEE 802.11e wireless LANs. In *Proceedings of the* 18^{th} *International Teletraffic Congress - ITC18*, Berlin, Germany, 2003.

[14] M. Barry, A. T. Campbell, and A. Veres. Distributed control algorithms for service differentiation in wireless packet networks. In *Proceedings IEEE INFOCOM 2001*, volume 1, pages 582–590, 2001.

[15] J.V.L. Beckers, I. Hendrawan, R.E. Kooij, and R.D. van der Mei. Generalized processor sharing models for internet access lines. In *Proceedings of IFIP Conference on Performance Modelling and Evaluation of ATM and IP networks*, pages 101–112, Budapest, 2001.

[16] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[17] S. Ben Fredj, T. Bonald, A. Proutière, G. Régnié, and J.W. Roberts. Statistical bandwidth sharing: a study of congestion at flow level. *SIGCOMM Computer Communication Review*, 31(4):111–122, 2001.

[18] F. Berggren and R. Litjens. Performance analysis of access selection and transmit diversity in multi-access networks. In *MobiCom '06: Proceedings of the* 12^{th} *Annual International Conference on Mobile Computing and Networking*, pages 251–261, New York, NY, U.S.A., 2006.

[19] S. Bhulai, G.J. Hoekstra, J.W. Bosman, and R.D. van der Mei. Dynamic traffic splitting to parallel wireless networks with partial information: A bayesian approach. *Performance Evaluation*, 69(1):41–52, 2012.

[20] S. Bhulai, G.J. Hoekstra, and R.D. van der Mei. Optimal concurrent access strategies in mobile communication networks. In *Proceedings of the* 22^{nd} *International Teletraffic Congress - ITC22*, Amsterdam, the Netherlands, 2010.

[21] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[22] S.C. Borst, O.J. Boxma, and N. Hegde. Sojourn times in finite-capacity processor-sharing queues. In *Proceedings NGI 2005 Conference*, 2005.

[23] S.C. Borst and D. Mitra. Virtual partitioning for robust resource sharing: computational techniques for heterogeneous traffic. *IEEE Journal on Selected Areas in Communications*, 16(5):668–678, 1998.

[24] S.C. Borst, R. Núñez-Queija, and A.P. Zwart. Sojourn time asymptotics in processor-sharing queues. *Queueing Systems*, 53(1-2):31–51, 2006.

[25] O.J. Boxma and H. Daduna. Sojourn times in queueing networks. In *Stochastic Analysis of Computer and Communication Systems, ed. H. Takagi (IFIP, North-Holland, Amsterdam, 1990)*, pages 401–450.

[26] O.J. Boxma and A.P. Zwart. Tails in scheduling. *SIGMETRICS Performance Evaluation Review*, 34(4):13–20, 2007.

[27] R. Braden. Requirements for internet hosts – communication layers. RFC 1122, Internet Engineering Task Force, October 1989.

[28] R.I. Brafman. A heuristic variable grid solution method for POMDPs. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 727–733, 1997.

[29] A.N. Burnetas and M.N. Katehakis. Optimal adaptive policies for Markov decision processes. *Mathematics of Operations Research*, 22:222–255, 1997.

[30] A.R. Cassandra. *Exact and Approximate Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Brown University, 1998.

[31] R. Chandra, P. Bahl, and P. Bahl. Multinet: Connecting to multiple IEEE 802.11 networks using a single wireless card. In *Proceedings IEEE INFOCOM 2004*, 2004.

[32] G.L. Choudhury and D.J. Houck. Combined queuing and activity network based modeling of sojourn time distributions in distributed telecommunication systems. In *Proceedings of the* 14^{th} *International Teletraffic Congress - ITC14*, pages 525–534, Antibes, France, 1994.

[33] E.G. Coffman, Jr., R.R. Muntz, and H. Trotter. Waiting time distributions for processor-sharing systems. *Journal of the ACM*, 17(1):123–130, 1970.

[34] D. Cox. Fundamental limitations on the data rate in wireless systems. *IEEE Communications Magazine*, 46(12):16–17, 2008.

[35] N.T. Dao and R.A. Malaney. Throughput performance of saturated 802.11g networks. In *AUSWIRELESS '07: Proceedings of the 2nd International Conference on Wireless Broadband and Ultra Wideband Communications*, 2007.

[36] J. Duncanson. Inverse multiplexing. *IEEE Communications Magazine*, 32(4):34–41, 1994.

[37] L. Flatto and S. Hahn. Two parallel queues created by arrivals with two demands. *SIAM Journal on Applied Mathematics*, 44(5):1041–1053, 1984.

[38] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot. Packet-level traffic measurements from the Sprint IP backbone. *IEEE Network*, 17:6–16, 2003.

[39] D. Gao, J. Cai, and K.N. Ngan. Admission control in IEEE 802.11e wireless LANs. *IEEE Network*, 19(4):6–13, 2005.

[40] M.S. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc., 2002.

[41] F. Guillemin, Ph. Robert, and A.P. Zwart. Tail asymptotics for processor sharing queues. *Advances in Applied Probability*, 36:525–543, 2004.

[42] V. Gupta, M. Harchol Balter, K. Sigman, and W. Whitt. Analysis of join-the-shortest-queue routing for web server farms. *Performance Evaluation*, 64(9-12):1062–1081, 2007.

[43] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Deployable multipath communication scheme with sufficient performance data distribution method. *Computer Communications*, 30(17):3285–3292, 2007.

[44] M. Hauskrecht. *Planning and Control in Stochastic Domains with Imperfect Information*. PhD thesis, Massachusetts Institute of Technology, 1997.

[45] A. Heindl and R. German. The impact of backoff, EIFS, and beacons on the performance of IEEE 802.11 wireless LANs. In *IPDS '00: Proceedings of the 4^{th} International Computer Performance and Dependability Symposium*, page 103, Washington, DC, U.S.A., 2000.

[46] O. Hernández-Lerma and J.B. Lasserre. *Discrete-Time Markov Control Processes: Basic Optimality Criteria*. Springer-Verlag, 1996.

[47] G.J. Hoekstra and F.J.M. Panken. Maintaining quality of service for wireless communications. Patent, June 2006. Application No. US 2006/450546 A.

[48] G.J. Hoekstra and F.J.M. Panken. Method of predicting transmission speed adaptations. Patent, August 2006. Application No. US 2006/463389 A.

[49] G.J. Hoekstra and F.J.M. Panken. On the efficient policing of HTTP traffic in WLANs. In *Proceedings Third ERCIM Workshop on eMobility*, pages 15–29, University of Twente, Enschede, The Netherlands, 2009.

[50] G.J. Hoekstra and R.D. van der Mei. On the processor sharing of file transfers in wireless LANs. In *Proceedings of the 69th IEEE Vehicular Technology Conference, VTC Spring 2009*, Barcelona, Spain, 2009.

[51] G.J. Hoekstra and R.D. van der Mei. Effective load for flow-level performance modelling of file transfers in wireless LANs. *Computer Communications*, 33(16):1972–1981, 2010.

[52] G.J. Hoekstra and R.D. van der Mei. On the Processor Sharing properties of file transfers in a WLAN testbed. In *Proceedings Internet-2010*, pages 36–41, Valencia, Spain, 2010.

[53] G.J. Hoekstra, R.D. van der Mei, and S. Bhulai. Optimal job splitting in parallel processor sharing queues. To appear in Stochastic Models, 28(1), 2012.

[54] G.J. Hoekstra, R.D. van der Mei, and J.W. Bosman. On comparing the performance of dynamic multi-network optimizations. In *Proceedings IEEE GLOBECOM 2010*, pages 1–5, Miami, U.S.A., 2010.

[55] G.J. Hoekstra, R.D. van der Mei, T. Dziedzicki, and J.W. Bosman. Efficient traffic splitting in parallel TCP-based networks: Modelling and experimental evaluation. Submitted for publication.

[56] G.J. Hoekstra, R.D. van der Mei, Y. Nazarathy, and A.P. Zwart. Optimal file splitting for wireless networks with concurrent access. *Lecture Notes in Computer Science*, 5894:189–203, 2009.

[57] R. Horak. *Telecommunications and Data Communications Handbook*. Wiley-Interscience, 2007.

[58] H.Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1):99–114, 2005.

[59] C.M. Huang and C.H. Tsai. The handover control mechanism for multipath transmission using stream control transmission protocol (SCTP). *Computer Communications*, 30(17):3239–3256, 2007.

[60] J.R. Iyengar. *End-to-end concurrent multipath transfer using transport layer multihoming*. PhD thesis, University of Delaware.

[61] J.R. Iyengar, P.D. Amer, and R. Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[62] M. Jaatun, I. Tøndel, F. Paint, T. Johannessen, J.C. Francis, and C. Duranton. Secure fast handover in an open broadband access network using kerberos-style tickets. In *Proceedings IFIP International Federation for Information Processing*, volume 201, pages 389–400, 2006.

[63] V. Jacobson. Congestion avoidance and control. *SIGCOMM Computer Communication Review*, 25(1):157–187, 1995.

[64] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. RFC 1323, Internet Engineering Task Force, May 1992.

[65] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D.F. Towsley. Inferring TCP connection characteristics through passive measurements. In *Proceedings IEEE INFOCOM 2004*, volume 3, pages 1582–1592, 2004.

[66] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D.F. Towsley. Measurement and classification of out-of-sequence packets in a tier-1 IP backbone. *IEEE/ACM Transactions on Networking*, 15(1):54–66, 2007.

[67] F.P. Kelly. *Reversibility and Stochastic Networks*. John Wiley and Sons Ltd., Chichester, 1979.

[68] P.B. Key, L. Massoulié, and D.F. Towsley. Combining multipath routing and congestion control for robustness. In *Proceedings Conference on Information Sciences and Systems*, 2006.

[69] P.B. Key, L. Massoulié, and D.F. Towsley. Path selection and multipath congestion control. In *Proceedings IEEE INFOCOM 2007*, pages 143–151, 2007.

[70] L. Kleinrock. Time-shared systems: a theoretical treatment. *Journal of the ACM*, 14(2):242–261, 1967.

[71] S.S. Ko and R.F. Serfozo. Response times in M/M/s fork-join networks. *Advances in Applied Probability*, 36(3):854–871, 2004.

[72] G.P. Koudouris, R. Agüero, E. Alexandri, J. Choque, K. Dimou, H.R. Karimi, H. Lederer, J. Sachs, and R. Sigle. Generic link layer functionality for multi-radio access networks. In *Proceedings* 14^{th} *IST Mobile and Wireless Communications Summit*, 2005.

[73] P.R. Kumar. A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, 23:329–380, 1985.

[74] M. Lelarge. Tail asymptotics for discrete event systems. In *Proceedings Valuetools '06:* 1^{st} *international conference on Performance evaluation methodolgies and tools*, pages 563–584, 2006.

[75] M. Lelarge. Packet reordering in networks with heavy-tailed delays. *Mathematical Methods of Operations Research*, 67(2):341–371, 2008.

[76] R. Litjens, F. Roijers, J.L. van den Berg, R.J. Boucherie, and M.J. Fleuren. Performance analysis of wireless LANs: an integrated packet/flow level approach. In *Proceedings of the* 18^{th} *International Teletraffic Congress - ITC18*, pages 931–940, Berlin, Germany, 2003.

[77] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Schaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, 1995.

[78] Z. Liu, N. Niclausse, and C. Jalpa-Villanueva. Traffic model and performance evaluation of web servers. *Performance Evaluation*, 46(2-3):77–100, 2001.

[79] J.A. Loeve. *Markov Decision Chains with Partial Information*. PhD thesis, Leiden University, 1995.

[80] W.S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

[81] J.C.S. Lui, R.R. Muntz, and D.F. Towsley. Computing performance bounds for fork-join queueing models. Technical report, The Chinese University of Hong Kong, 1994.

[82] P. Mahasukhon, M. Hempel, S. Ci, and H. Sharif. Comparison of throughput performance for the IEEE 802.11a and 802.11g networks. In *Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA 2007), Niagara Falls, Canada*, pages 792–799, 2007.

[83] A. Markopoulou, F. Tobagi, and M. Karam. Loss and delay measurements of internet backbones. *Computer Communications*, 29(10):1590–1604, 2006.

[84] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018, Internet Engineering Task Force, 1996.

[85] K. Medepalli, P. Gopalakrishnan, D. Famolari, and T. Kodama. Voice capacity of IEEE 802.11b, 802.11a and 802.11g wireless LANs. In *Proceedings IEEE GLOBECOM 2004*, volume 3, pages 1549–1553, 2004.

[86] Microsoft Corporation. Next Generation TCP/IP Protocols and Networking Components, January 2008. `http://technet.microsoft.com/en-us/library/cc754287.aspx`.

[87] D. Miorandi, A.A. Kherani, and E. Altman. A queueing model for HTTP traffic over IEEE 802.11 WLANs. *Computer Networks*, 50(1):63–79, 2006.

[88] T. Miyano, S. Otsuki, M. Umeuchi, and M. Ogasawara. Admission and traffic control techniques for WLANs. *NTT Technical Review*, 5(11), 2007.

[89] G.E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28:1–16, 1982.

[90] M. Ogasawara, M. Umeuchi, K. Kawamura, T. Miyano, S. Otsuki, K. Nagata, and T. Hiraguri. Overview of QoS control techniques for wireless local area networks. *IEEE Communications Magazine*, 5(11):1–5, 2007.

[91] OPNET Technologies Inc. OPNET modeler. `http://www.opnet.com/solutions/network_rd/modeler.html`, November 2011.

[92] F.J.M. Panken and G.J. Hoekstra. On regulating traffic in shared wireless access media. Submitted for publication.

[93] F.J.M. Panken and G.J. Hoekstra. Multi-service traffic profiles to realise and maintain QoS guarantees in wireless LANs. *Computer Communications*, 32(6):1022–1033, 2009.

[94] F.J.M. Panken, G.J. Hoekstra, D. Barankanira, J.C. Francis, R. Schwendener, O. Grondalen, and M.G. Jaatun. Extending 3G/WiMAX networks and services through residential access capacity. *IEEE Communications Magazine*, 45(12):62–69, 2007.

[95] F.J.M. Panken, G.J. Hoekstra, and T. Van der Gaast. Resource allocation and guarantees for real-time applications in WLANs. *Telektronikk*, (3):125–134, 2006.

[96] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[97] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1088–1094, 1995.

[98] D. Pong and T. Moors. Call admission control for IEEE 802.11 contention access mechanism. In *Proceedings IEEE GLOBECOM 2003*, volume 1, pages 174–178, San Francisco, U.S.A., 2003.

[99] J. Postel. Transmission control protocol - darpa internet program protocol specification. RFC 793, Internet Engineering Task Force, September 1981.

[100] J. Postel and J. Reynolds. File transfer protocol (FTP). RFC 959, Internet Engineering Task Force, October 1985.

[101] J. Postel and J. Reynolds. Standard for the transmission of IP datagrams over IEEE 802 networks. RFC 1042, Internet Engineering Task Force, February 1988.

[102] ProFTPD Project. Professional FTP Daemon, November 2011. `http://www.proftpd.org/`.

[103] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[104] A. Riedl, T. Bauschert, M. Perske, and A. Probst. Investigation of the M/G/R Processor Sharing Model for Dimensioning of IP Access Networks with Elastic Traffic. In *Proceedings of the First Polish-German Teletraffic Symposium PGTS 2000*, Dresden, 2000.

[105] J.W. Roberts. A survey on statistical bandwidth sharing. *Computer Networks*, 45(3):319–332, 2004.

[106] F. Roijers, J.L. van den Berg, and X. Fang. Analytical modelling of TCP file transfer times over 802.11 wireless LANs. In *Proceedings of the* 19^{th} *International Teletraffic Congress - ITC19*, Beijing, China, 2005.

[107] K.W. Ross. *Multirate Loss Models for Broadband Telecommunication Networks*. Springer-Berlin, 1995.

[108] T. Sakurai and S. Hanley. Modelling TCP flows over an 802.11 wireless LAN. In *Proceedings of European Wireless Conference*, 2005.

[109] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms. RFC 2001, Internet Engineering Task Force, 1997.

[110] W.R. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional, New York, NY, U.S.A., fourth edition, 1993.

[111] R. Stewart. Stream control transmission protocol. RFC 2960, Internet Engineering Task Force, October 2000.

[112] Y. Tian, K. Xu, and N. Ansari. TCP in wireless environments: problems and solutions. *IEEE Communications Magazine*, 43(3):27–32, 2005.

[113] J.S. Turner. New directions in communications (or which way to the information age?). *IEEE Communications Magazine*, 40(5):50–57, 1986.

[114] J.L. van den Berg. *Sojourn Times in Feedback and Processor Sharing Queues*. PhD thesis, University of Utrecht, 1990.

[115] J.L. van den Berg and O.J. Boxma. The M/G/1 queue with processor sharing and its relation to a feedback queue. *Queueing Systems*, 9(4):365–402, 1991.

[116] T. van der Gaast, G.J. Hoekstra, F.J.M. Panken, and J. van Bemmel. Method for assigning uplink and/or downlink capacities based on available capacity. Patent, August 2005. Application No. US 2005/204194 A.

[117] K.M. van Hee. *Bayesian Control of Markov Chains*. PhD thesis, Technical University of Eindhoven, 1978.

[118] A. Veres, A.T. Campbell, M. Barry, and L.H. Sun. Supporting service differentiation in wireless packet networks using distributed control. *IEEE Journal on Selected Areas in Communications*, 19:2094–2104, 2002.

[119] C.C. White III. A survey of solution techniques for the partially observed Markov decision process. *Annals of Operations Research*, 32:215–230, 1991.

[120] Y. Wu, C. Williamson, and J. Luo. On processor sharing and its applications to cellular data network provisioning. *Performance Evaluation*, 64(9-12):892–908, 2007.

[121] Y. Xiao and H. Li. Evaluation of distributed admission control for the IEEE 802.11e EDCA. *IEEE Communications Magazine*, 42(9):S20–S24, 2004.

[122] Y. Xiao and H. Li. Voice and video transmissions with global data parameter control for the IEEE 802.11e enhance distributed channel access. *IEEE Transactions on Parallel Distributed Systems*, 15:1041–1053, 2004.

[123] L. Zhang and S. Zeadally. HARMONICA: Enhanced QoS Support with Admission Control for IEEE 802.11 Contention-based Access. In *Proceedings of the* 10^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 64–73, Washington, DC, U.S.A., 2004.

[124] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 99–112, Berkeley, CA, U.S.A., 2004.

[125] N.L. Zhang and W. Liu. Region-based approximations for planning in stochastic domains. In *Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 472–480, 1997.

[126] H. Zhu, M. Li, I. Chlamtac, and B. Prabhakaran. A survey of quality of service in IEEE 802.11 networks. *IEEE Wireless Communications*, 11(4):6–14, 2004.

[127] A.P. Zwart. Sojourn times in a multiclass processor sharing queue. In *Proceedings of the* 16^{th} *International Teletraffic Congress - ITC16, eds. P. Key, D. Smith (North-Holland, Amsterdam)*, pages 335–344, Edinburgh, UK, 1999.

[128] A.P. Zwart and O.J. Boxma. Sojourn time asymptotics in the M/G/1 processor sharing queue. *Queueing Systems*, 35(1/4):141–166, 2000.