Utility-based Optimisation of Resource Allocation for Wireless Networks

by

Calin Curescu

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2005
Abstract

Wireless communication networks are facing a paradigm shift. New generations of cellular and ad hoc networks are finding new application areas, be it in commercial communication, for military use or disaster management.

From providing only voice communications, wireless networks aim to provide a wide range of services in which soft real-time, high priority critical data, and best effort connections seamlessly integrate. Some of these applications and services have firm resource requirements in order to function properly (e.g. videoconferences), others are flexible enough to adapt to whatever is available (e.g. FTP). Different connections might have different importance levels, and should be treated accordingly. Providing differentiation and resource assurance is often referred to as providing quality of service (QoS).

In this thesis we study how novel resource allocation algorithms can improve the offered QoS of dynamic, unpredictable, and resource constrained distributed systems, such as a wireless network, during periods of overload.

We propose and evaluate several bandwidth allocation schemes in the context of cellular, hybrid and pure ad hoc networks. Acceptable quality levels for a connection are specified using resource-utility functions, and our allocation aims to maximise accumulated system-wide utility. To keep allocation optimal in this changing environment, we need to periodically reallocate resources. The novelty of our approach is that we have augmented the utility function model by identifying and classifying the way realloca-
tions affect the utility of different application classes. We modify the initial utility functions at runtime, such that connections become comparable regardless of their flexibility to reallocations or age-related importance.

Another contribution is a combined utility/price-based bandwidth allocation and routing scheme for ad hoc networks. First we cast the problem of utility maximisation in a linear programming form. Then we propose a novel distributed allocation algorithm, where every flow bids for resources on the end-to-end path depending on the resource “shadow price”, and the flow’s “utility efficiency”. Our periodic (re)allocation algorithms represent an iterative process that both adapts to changes in the network, and recalculates and improves the estimation of resource shadow prices.

Finally, problems connected to allocation optimisation, such as modelling non-critical resources as costs, or using feedback to adapt to uncertainties in resource usage and availability, are addressed.
Acknowledgements

Foremost I would like to thank my supervisor Simin Nadjm-Tehrani for her constant support, guidance, and collaboration during these five years. Her enthusiasm in addressing new challenges and her goal oriented approach are examples for me to follow. I would also like to thank Jörgen Hansson for the no-nonsense discussions and for his insightful comments on my work and thesis.

Working together with other people provided sparks for new ideas and finality for others. I would like to thank Teresa A. Dahlberg and all my collaborators from academia and industry for their inputs and the rewarding collaboration.

A big thanks goes to my colleagues at Linköping University, especially to the RTSLAB members, for the lively discussions and debates, and the pleasant work environment. My thanks also to the administrative and technical staff for their prompt support. Anne Moe has always been helpful, even when I left things for the last minute.

CUGS, the Swedish National Graduate School in Computer Science provided not only funds for my research but also opportunities for peer meetings and coursework. ECSEL, the Excellence Center in Computer Science and Systems Engineering in Linköping, provided funding for my first research year.

I wish to salute all my friends, for making my life more enjoyable, and for supporting me in many different ways, especially Sorin for showing me this path. Last but not least, I am deeply grateful to Deka and my family for their love and constant support.

Călin Curescu
# Table of Contents

1 Introduction
   1.1 Motivation ............................................. 9
   1.2 Problem description ................................. 11
   1.3 Contributions ....................................... 13
   1.4 Thesis outline ...................................... 15
   1.5 List of Publications ................................. 16

2 QoS Preliminaries ....................................... 19
   2.1 CPU allocation in real-time systems .............. 20
   2.2 QoS in the Internet ................................. 22
   2.3 Characteristics of a QoS architecture ............ 25
       2.3.1 QoS specification .............................. 26
       2.3.2 QoS enforcement mechanisms ................. 27

3 Utility model ............................................ 31
   3.1 Application adaptation .............................. 31
   3.2 Utility optimisation and fairness ................. 33
   3.3 Application utility .................................. 34
   3.4 Q-RAM utility model ................................ 35
   3.5 Optimising single resource allocation .............. 38
   3.6 Utility setup for network traffic ................. 42

4 Adaptive reallocation .................................. 49
   4.1 Overview .............................................. 49
   4.2 Related work ......................................... 52
# TABLE OF CONTENTS

4.3 Application area ........................................... 54
4.4 Reallocation consequences ................................. 58
  4.4.1 Flexibility classes .................................. 60
  4.4.2 Drop penalty ....................................... 61
  4.4.3 Adaptation time .................................... 61
4.5 Dynamic reallocation ....................................... 62
  4.5.1 Age and class influence ............................. 64
  4.5.2 Drop penalty influence ............................ 67
  4.5.3 Adaptation time influence .......................... 68
  4.5.4 Algorithm overview ................................ 69
4.6 Evaluation setup .......................................... 71
  4.6.1 RBBS description ................................ 72
  4.6.2 Traffic and simulation parameters ................. 73
4.7 Evaluation results ........................................ 74
  4.7.1 Comparison with basic maximisation ............... 75
  4.7.2 Comparison with RBBS ............................. 76
  4.7.3 Effects of convex hull approximation ............ 77
  4.7.4 QoS per application group ........................ 78
  4.7.5 Choice of performance metric ..................... 78
  4.7.6 Connection duration estimation ................... 80
  4.7.7 Complexity considerations ......................... 81
4.8 Reallocation overhead ...................................... 82
  4.8.1 Reallocation control ............................... 84
4.9 Summary ................................................... 87

5 Non-critical resources as cost ............................... 89
  5.1 Overview ................................................. 90
  5.2 Related work ........................................... 91
  5.3 Allocation in hybrid networks .......................... 93
    5.3.1 System model .................................... 93
    5.3.2 Optimal bandwidth allocation .................... 95
    5.3.3 Linear programming formulation ................. 97
    5.3.4 Hybrid-heuristic algorithm ...................... 100
  5.4 Evaluation setup ....................................... 101
  5.5 Experimental results ................................... 102
### Table of Contents

5.5.1 Accumulated utility as performance .................................... 103  
5.5.2 QoS preservation ......................................................... 106  
5.5.3 Time complexity ......................................................... 108  
5.5.4 Cost influence ............................................................ 109  
5.6 Summary ................................................................. 109  

6 Price-based allocation .................................................. 111  
6.1 Overview ................................................................. 112  
6.2 Related work ............................................................. 113  
6.3 Problem formulation ...................................................... 115  
6.3.1 Network model ......................................................... 115  
6.3.2 Linear programming formulation .................................... 118  
6.3.3 Dual form and optimal solution properties ....................... 122  
6.4 Distributed routing and allocation .................................... 124  
6.4.1 Bid construction ....................................................... 125  
6.4.2 Independent allocation .............................................. 125  
6.4.3 Discussions ............................................................. 126  
6.4.4 QoS routing ............................................................. 129  
6.4.5 Mobility and clique construction ................................... 130  
6.5 Simulation and results ................................................. 131  
6.5.1 Evaluation setup ...................................................... 131  
6.5.2 Comparison of allocation schemes .................................. 132  
6.5.3 Experimental results ................................................ 134  
6.6 Summary ................................................................. 139  

7 Dealing with Uncertainties ............................................. 141  
7.1 Adaptive load control .................................................... 142  
7.2 Related work ............................................................. 143  
7.3 Load control in 3G networks .......................................... 145  
7.4 Algorithm design ........................................................ 148  
7.5 Evaluation setup ........................................................ 152  
7.6 Simulation results ....................................................... 154  
7.7 Summary ................................................................. 156
# TABLE OF CONTENTS

8 Conclusions and Future Work 161

8.1 Conclusions ....................... 161
8.2 Future work ...................... 165

Bibliography 167
List of Figures

3.1 R-U function quantisation .................................. 38
3.2 The convex hull optimisation algorithm .................. 40
3.3 Optimal allocation order ..................................... 41
3.4 Utility function shape for application group 1 ....... 45
3.5 Utility function shape for application group 2 ....... 45
3.6 Utility function shape for application group 3 ....... 46
3.7 Utility function shape for application group 4 and 6 47
3.8 Utility function shape for application group 5 ....... 47

4.1 The UMTS Radio Access Network ......................... 55
4.2 Replacement opportunity ..................................... 65
4.3 Age modification for class I and II, with $t_{i}^{\text{age}} = 5$, $t_{i}^{\text{max}} = 10$, and actual bandwidth $b_i = 4$ ....... 66
4.4 Class II drop modification with $P_{i}^{\text{drop}} = 8$, $t_{i}^{\text{age}} = 5$, $t_{i}^{\text{max}} = 10$, and actual bandwidth $b_i = 4$ ....... 68
4.5 Class III adaptation modification with $A_i = 5$, $I_i = 4$, $P_{i}^{\text{adapt}} = 2$, and actual bandwidth $b_i = 4$ ....... 69
4.6 The TARA (re)allocation algorithm ....................... 70
4.7 The TARA utility accounting algorithm .................. 71
4.8 Accumulated utility .......................................... 75
4.9 Connection blocking probability ............................. 79
4.10 Connection dropping probability .......................... 80
4.11 Differences due to estimated duration .................... 81
4.12 Number of reallocations comparison ...................... 83
4.13 Reallocation control algorithm ............................ 85
   View (a) .................................. 86
4.15 Performance of reallocation minimisation algorithm.
   View (b) .................................. 87

5.1 Hybrid network ................................ 95
5.2 Hybrid network - condensed ...................... 95
5.3 Segment of a utility function .................... 97
5.4 Utility performance for the hot-spots scenario .... 103
5.5 Utility originating in an overloaded cell .......... 105
5.6 Utility performance for the BS crash scenario ... 106
5.7 Utility performance for the balanced load scenario 107
5.8 Dependency on the cost parameter ................. 108

6.1 Network example .................................. 117
6.2 Link contention graph for network example ....... 118
6.3 Linear segments of a convex hull .................. 119
6.4 The distributed allocation algorithm ............... 127
6.5 The clique reconstruction algorithm ............... 130
6.6 Ad hoc network snapshot .......................... 133
6.7 Utility for rt-mix traffic .......................... 135
6.8 Utility for flexible traffic ......................... 136
6.9 Utility as function of traffic load ................. 139

7.1 Schematic overview of processor protection ....... 147
7.2 Schematic description of the architecture ......... 149
7.3 The Pool algorithm ............................... 151
7.4 The “www-browsing” session ....................... 152
7.5 Leaky-bucket admission ............................ 153
7.6 Cumulative overload beyond the desired threshold 154
7.7 Load comparison .................................... 155
7.8 Behaviour with a 50/50 differentiation policy ...... 157
List of Tables

3.1 A traffic mix setup ...................... 43
4.1 Notation summary .......................... 64
4.2 Approximation bound for \textit{convex\_hull\_opt} ........ 77
4.3 Statistics per application group at load 2.42 and 13 78
5.1 Statistics per application group at load 2.42 . . . . 107
5.2 Computational complexity of the algorithms . . . . 108
6.1 Flow-resource usage for network example ............. 118
6.2 Traffic mix used in the ad hoc experiments ......... 132
6.3 Bandwidth usage of adhoc-TARA compared to LP 136
6.4 Approximation bound for LP algorithm ............... 137
6.5 Statistics at application group level ................. 138
6.6 Average running time of allocation decision .......... 138
Chapter 1

Introduction

1.1 Motivation

Although the deployment of third generation (3G) wireless communication networks has not been as successful as predicted during the 90’s, ubiquitous broadband communication is still actively pushed forward by the desire for multimedia communication. Wireless networks of third (3G) and future generations are different from second generation (2G) networks, e.g. the Global System for Mobile Communications (GSM), in several ways.

First, the bandwidth of the radio link is much larger than in a second generation network. For example, the Wide-band Code Division Multiple Access (WCDMA) technology enables communication speeds of 2 Mbps compared to 14-64 Kbps in GSM. This enables besides voice communication, many other types of more advanced multimedia communications and data services. Second, using the packet-switched transmission paradigm means that network resources can be allocated and accounted with a much finer granularity, enabling the use of intelligent resource allocation algorithms.

A strong complement to cellular networks are wireless mobile ad hoc networks. Ad hoc networks are formed by wireless nodes that move freely and have no fixed infrastructure. Each node in the network may act as a router for other nodes, and flows follow
a multi-hop path from a source to a destination. The flexibility that comes with lack of fixed infrastructure makes them ideal for novel scenarios, including cooperative environments, defence applications and disaster management. Both network types aim to provide a wide range of services in which soft real-time (multimedia), and high priority critical data seamlessly integrate.

Resource allocation problems with the aim of guaranteeing a certain level of service have been studied in the Internet community for many years. But do the results apply also to wireless networks? As a matter of fact, there are several differences which make the resource allocation problem more complicated.

First, there is the nature of the wireless channel which entails the following:

- The limited spectrum available. Wireless networks share the same bandwidth and location-based interference can greatly reduce the communication capacity. Overload situations cannot be solved by over-provisioning; in wireline networks bandwidth can be increased as simply as adding another wire.

- The fluctuating nature of the wireless channel. Due to fading and interference, the bandwidth of the radio link may vary greatly during the stay of the user in one area of the radio network.

- User mobility. In cellular networks, at handover the user enters another cell where resource availability might be radically different. In ad hoc networks the effects of mobility are even more radical. Multi-hop routes can quickly break, and capacity can quickly change with nodes entering and leaving the interference area of other nodes.

The second difference is that most nodes of such a network are autonomous, embedded systems, with limited upgrade possibility, and sensitive to issues like power consumption, size, heat generation, and processing capabilities.
On top of this, these networks usually provide an open environment. That is, users might join and leave, or communicate only on special occasions. Thus, the load is hard to predict and average load might differ greatly from peak load. Moreover, network utilisation patterns are expected to be as unpredictable and bursty as in the Internet. We can easily imagine some examples where overloads are unavoidable: A highway jam, the ending of a football game, a disaster site, a “breaking news” everyone is interested in.

1.2 Problem description

The previous characteristics show that wireless packet-switched networks are very susceptible to overloads. An overload situation is occurring when the applications in the network require more resources than the system has available. At overload, connections belonging to the applications active in the network might be delayed indefinitely, or even be rejected, or dropped.

The aim of our work is to study how novel resource allocation algorithms can improve the behaviour of dynamic, unpredictable distributed systems, such as wireless networks, during periods of overload. To this end we next present the factors on which the performance of the system depends.

Some of the applications and services provided on these networks, such as multimedia communications, multi-player games or monitoring services, require dependable performance guarantees from the network in order to function properly. This can be enforced by allocating the needed resources at the right time. Applications requiring performance guarantees share the link with ones that require less or no performance guarantees, such as e-mail or file transfer. Thus the system has to differentiate between applications and treat them accordingly. Moreover, some customers are prepared to pay more to be treated preferentially, so differentiation is not only dependent on the application type but also on its user. The capability to provide resource assurance and differentiation is often referred to as provision of quality of service (QoS) [1].
One more factor to be taken into consideration is the adaptability of the applications themselves. Many applications can adapt to different resource amounts even if it results in lower performance. For example, different compression mechanisms with different bandwidth requirements and output quality can be used for transmitting a video over the network. The flexibility with regard to resource needs can be exploited when allocating and reallocating resources, for increasing the performance of the system and/or for providing graceful degradation for the applications.

Dealing with resource assurance can be quickly solved through resource reservation. However, there is an inherent conflict between reservation and adaptation, and without the latter the allocation may quickly become suboptimal. Rigid reservation is infeasible in such a dynamic environment. New methods, that constantly update resource allocations for an optimised system performance, are needed. At the same time the allocation algorithms have to take into consideration the resource assurances required by the applications.

Much research concentrated on enforcing a certain QoS allocation. However, another important part of the equation is deciding what QoS to be offered to the participants in order to improve the general performance of the system. Global policies and algorithms that optimise resource allocation are needed. Due to the dynamic nature of wireless networks, they should have light computational and signalling overhead, should be adaptive to the quickly changing environment, and should work in a distributed manner (in the case of multi-hop networks).

This thesis contributes to QoS aware resource allocation by considering optimised methods for allocating the bandwidth of the wireless link as a bottleneck resource. Connected problems such as modelling non-critical resources and employing feedback control for dealing with uncertainty are also addressed.
1.3 Contributions

In the context of a general resource allocation framework for dynamic, open systems we have developed several algorithms aimed to optimise bandwidth allocation in wireless networks. We assume that acceptable quality levels for the different end-to-end connections are quantitatively specified using discrete resource dependent utility functions. We propose several allocation schemes that use these utility functions for allocating and reallocating bandwidth to connections, aiming to maximise the accumulated utility of the system. The contributions of the thesis are the following:

- **An adaptive utility-based bandwidth allocation and reallocation scheme for cellular networks.** The utility functions of the connections are used to steer the allocation, with the aim to optimise the overall system utility (the sum of the utilities gained by all the connections). To keep the dynamic system optimised we periodically reevaluate all allocations and perform the necessary reallocations. The novelty in our scheme is that we take into account the effects that reallocations have on the accumulated utility of the different connections. We have evaluated our new approach by comparing it with a recent adaptive allocation scheme [2]. Our time-aware resource allocation (TARA) scheme is presented in Chapter 4.

- **An extension of the utility function model that takes into account the effects of reallocations.** Some applications need strong resource assurance, others do not. Utility functions however, do not contain information regarding the sensitivity of applications to reallocation. We have augmented the utility model by identifying and classifying the way reallocations affect the utility of different application flexibility classes. We have built algorithms for modifying the initial utility functions at runtime, such that connections become comparable regardless of their flexibility class or age. Thus, adaptation and resource assurance are combined in a consistent manner, and our system supports real-time applications.
alongside best-effort ones. This is covered in Sections 4.4, 4.5 of the thesis.

- **An allocation algorithm that can trade off utility for a lower reallocation-related overhead.** Reallocations can create quite a computational and signalling overhead in the network. To alleviate this we show how to skip over some of the allocation changes and significantly decrease reallocation overhead while keeping a high system utility level. This is treated in Section 4.8.

- **An extension of the TARA scheme for hybrid wireless networks, modelling costs and incentives.** An “ad hoc over cellular” hybrid network extends the cellular network with ad hoc paths. The bottleneck resource is the bandwidth of the cellular link (ad hoc links are considered unconstrained in comparison), however, there is the possibility to use different paths to connect to different base stations. Thus we have extended the TARA scheme with a distributed utility-based path choice algorithm. We also model non-critical resources as a cost function (together with proportional incentives to be paid to the relaying users) and include them in the optimisation problem. Utility-wise, the scheme compares favourably with a pure cellular scheme, it provides load balancing among base stations and fault tolerance in the case of coverage failure. Chapter 5 presents this work.

- **A combined utility/price-based bandwidth allocation and routing scheme for ad hoc networks.** In this case, we study bandwidth allocation in a pure multi-hop environment. We first show that the bandwidth allocation problem in a multi-hop wireless network can be formulated as linear programming (LP) maximisation problem. While centralised and infeasible for online allocation, the LP solution provides an upper baseline for comparisons. Then we propose a novel distributed allocation algorithm, where every
flow “bids” for resources on the end-to-end path of the connection, depending on the resource “shadow price”, and the flow’s “utility efficiency”. Using the periodic (re)allocation, our scheme both adapts to changes in the network, and also recalculates and improves our estimation of the resource shadow prices. For deciding the routes we use a new type of shortest path first algorithm, where the shadow prices are used a natural distance metric. Experimental results show a very good behaviour of our distributed allocation algorithm, that is within 10% of the behaviour of the centralised, optimal LP solution. This contribution is presented in Chapter 6.

- Feedback methods for dealing with uncertainties.
  Open loop methods are insufficient when dealing with unexpected usage patterns or inaccurate resource consumption. As an example where feedback techniques can be used, we propose a feedback control algorithm to control the CPU load in a radio network controller (RNC). We provide an overload protection scheme for the processor while rejecting as few tasks as possible, a problem that is motivated by a real concern of the industry. We also incorporate task and user type differentiation, to support the QoS guarantees decided by a higher-level resource allocation policy. This is presented in Chapter 7.

1.4 Thesis outline

The thesis is organised as follows. In Chapter 2 background work on resource allocation and general characteristics of a QoS architecture are presented. Chapter 3 we present the utility model together with the utility functions for the traffic used in our experiments. Chapter 4 presents and evaluates the reallocation-aware utility optimisation scheme applied in the context of a cellular network. In Chapter 5 we extend the bandwidth allocation scheme to hybrid (ad hoc over cellular) networks, and show its load balancing
and fault-tolerant characteristics. In this context we also include the cost modelling of non-critical resources. For a fully ad hoc network, in Chapter 6 we present our combined bandwidth allocation and routing scheme based on shadow prices. In Chapter 7 we consider the advantages of using feedback in dealing with allocation uncertainties. Finally Chapter 8 summarises the conclusions and outlines future work.

1.5 List of Publications

The work presented in this thesis has been published in the following papers.


1.5. LIST OF PUBLICATIONS

Chapter 2

QoS Preliminaries

Resource allocation is an intensive area of research, both in the real-time (RT) community where it concerns the processor (CPU) as the main resource and in the networks community where it concerns network links.

This chapter aims to provide a perspective on the RT and networking areas as a larger context to which this work belongs.

Providing performance guarantees to distributed applications is actually an end-to-end issue; beginning at the source host, continuing over the network, and ending at the destination host. Thus several quality of service (QoS) architectures have been proposed, for managing QoS throughout all the architectural layers of a distributed system in a unified manner [3].

The chapter first provides a short survey of relevant work on CPU allocation, followed by reviews of the mechanisms for providing QoS in the Internet. Finally it presents the main characteristics of an end to end QoS architecture.

Note that the chapter is not intended to cover related works. Rather, when detailing our algorithms, the related works for each of the allocation schemes will be covered in the relevant chapters.
2.1 CPU allocation in real-time systems

A major concern of research on real-time systems is predictable scheduling under time constraints [4]. Hence CPU time is the main resource, with contention for other resource types being treated as “task execution blocking” [5].

Early real-time research was mainly concerned with mission-critical “hard real-time” tasks, executing in isolated, embedded environments. All the tasks were regarded as equally important, and a missed deadline meant system failure. To achieve this, static priority-based scheduling, such as rate monotonic (RM), and dynamic priority scheduling such as earliest deadline first (EDF) were proposed [6]. To provide scheduling guarantees, schedulability analysis must be performed a priori, and the worst case execution time of tasks has to be considered, leading to low utilisation under normal work conditions. The performance of the scheduling policy can be measured by the type of task sets it can optimally schedule or by the computational complexity it requires to create a schedule [4].

Going away from the strict hard real-time model, several variations and extensions have been proposed, like dealing with resource and task dependencies, scheduling tasks on multiple processors and including aperiodic and soft-realtime tasks. Tasks are regarded soft real-time if the consequences of missing the deadline are not critical.

Soft, aperiodic tasks can be scheduled together with hard-real-time tasks with the help of “servers”, which are periodic tasks whose purpose is to service aperiodic requests as soon as possible. There exist several flavours for both RM and EDF scheduling policies, and their performance can be compared by using the average response times achieved by the aperiodic tasks [4].

Multimedia applications (e.g. video or audio decoding) belong to the category of periodic soft real-time tasks, where infrequently missing a deadline is not critical. For these task types, providing timeliness guarantees using worst-case execution time would be too wasteful. Thus, schedulability analysis can be based on aver-
age computation times and expected miss ratio. To better integrate such tasks in real-time systems, share-based CPU-allocation methods have been proposed. Capacity Reserves [7] and Constant Bandwidth Server (CBS) [8] are two examples of these methods. Abeni and Buttazzo provide a comparison between CBS and several Weighted Fair Queuing (WFQ) strategies, the latter being used in network flow scheduling [9].

A new concept of dynamic RT-systems that have to work in unpredictable environments (open, distributed systems), in which load patterns are not known in advance was introduced by Stankovic et al. [10, 11]. In such unpredictable environments, overload situations may appear, and have to be dealt with. When this happens, tasks have to be rejected, dropped, or partially executed.

To make such decisions, differentiation mechanisms between tasks have to be considered. A worth value can be associated to tasks, and used during overloads for rejecting lower value tasks, for example as in the Robust Earliest Deadline (RED) scheduling policy [12]. Moreover, by using a deadline dependent value function, the value of different types of firm and soft real-time tasks can be expressed (as depending on how the completion time differs from the deadline). Jensen, Locke et al. [13, 14] introduced such “value functions” and used them for scheduling decisions during periods of overload. Thus, by using values attached to the different tasks, the performance of the scheduling algorithm can be quantitatively measured [15, 4].

An approach where tasks are not rejected, but partially executed constitutes the imprecise computations model [16, 17]. This model splits a task in a mandatory and an optional part. Depending on how much of the optional part is completed before the deadline, an error is calculated. The total error, which is the weighted sum of the errors of each task, can be used as a performance metric or as an optimisation criterion for a scheduler.

Another concern is directed on how to exploit the adaptability of the applications to the amount of computation time they receive. In the works of Abdelzaher et al. [18] and Brandt et al. [19] several QoS levels specify the utility of the application for dif-
ferent levels of CPU consumption. The aim of the systems is to accommodate as many of the arriving tasks as possible. Therefore, the QoS level of the tasks is lowered during periods of high load, and raised back when the load is light. Heuristic algorithms are used for choosing which tasks to degrade/upgrade first. Lee et al. [20, 21] propose a general theoretical framework where different levels of QoS depend on different resource allocations. Then again, the utility of an application depends on different QoS aspects. These two dependencies are combined when constructing resource dependent utility functions. Several resource allocation algorithms that have as input the applications’ resource-utility functions, and aim to maximise total system utility are proposed and evaluated in their work.

Another approach to adaptation proposes feedback control for highly dynamic environments, where task execution times and task arrival times might not be known in advance. The systems rely on a feedback loop to control the load on the processor [19, 22, 23, 24]. For instance [19, 24], if the system is underloaded the feedback is used to decrease CPU idle-time to zero. When the system is overloaded it is used to decrease deadline misses to zero. The system adapts by raising or decreasing the CPU allocation to tasks and thus changing the QoS of the tasks. While some of the feedback systems are based on heuristic algorithms [19, 25], methods based on control theory are becoming quite popular due to the theoretical analysis framework available [22, 23, 24, 26].

2.2 QoS in the Internet

In this section we provide a short overview of the mechanisms developed for QoS guarantees in IP networks. Research in this area covers a massive volume of results, but for our purposes it suffices to bring into focus important developments that are relevant as a perspective for our work. Components for an emerging QoS-aware Internet are also presented in several overviews [27, 28, 29], and more recent advances in the area are presented by Firoiu et al. [30].
The QoS received by the different applications or services from the network depends on how the generated streams of related packets (flows) are treated. Thus, QoS provisioning in IP networks might be roughly divided in the following categories. First, QoS frameworks describe control mechanisms for:

- establishing the path (routing),
- performing admission control,
- reserving resources, and
- taking corrective measures in case of congestion.

Then, at the path level, different enforcement mechanisms are used to ensure that flows behave according to their specifications and to provide the guaranteed service decided by the higher level control mechanisms.

Two frameworks have emerged as principal IP QoS architectures. Integrated Services (IntServ) [31] is a QoS framework where QoS is provided on a per-flow basis. A dynamic resource reservation protocol (RSVP) [32] is used by IntServ. RSVP employs a receiver-initiated reservation style. When the destination host receives a PATH message from the source host, it sends a RESV (reserve) message to all intermediary routers. If they accept the reservation request, buffer space and link bandwidth is reserved. A “soft state” describing the flow is installed on the router, which has to be constantly refreshed by end-hosts. While robust, RSVP creates some message overhead. However, the biggest overhead is created by routers maintaining a separate queue for each flow. To alleviate these problems, Differentiated Services (DiffServ) [33] has been proposed. The framework offers a per-aggregate-class QoS. No state is maintained in the router, the class being specified in the “Type of Service” byte of the IPv4 header. At each node the packet receives a per-hop behaviour according to its class. Packet classification and traffic conditioning is done only at network boundaries. DiffServ does not depend on a resource reservation protocol, although using one could improve QoS guarantees. Thus, per-flow guarantees of IntServ are traded for efficiency and scalability in DiffServ.
The process a packet might go through at path level is described next. When a packet arrives at a router a “packet classifier” identifies to which QoS class or flow the packet belongs to. A “meter” is measuring if the flow is behaving according to the specifications. Flows are usually specified using token bucket [34], with parameters such as maximum service rate, maximum burst size, peak rate, minimum and maximum packet size. If a packet does not conform to the specification it is either dropped (“traffic policing”) or delayed in a “shaper”. Alternatively it is “marked” as “out of profile”, to be identified by later processing.

Once classified, the packet is put in a queue, waiting to be scheduled for sending on the output link. Queue size and congestion control disciplines determine the degradation suffered by the flow during congestion.

Queuing delay is mainly determined by the scheduling policy employed. The scheduling policy may also determine rate guarantees to the flow. Scheduling disciplines roughly belong to the following classes.

- First Come First Serve (FCFS) offers neither differentiation nor performance guarantees.

- Weighted Fair Queuing (WFQ) schedules packets in a weight-ed round-robin fashion, the weights representing reserved link rates. Performance guarantees are offered for both delay and rate, however, the delay bound is closely linked to the rate [35].

- Earliest Deadline First (EDF), packets are scheduled according to deadlines; together with traffic shapers, EDF can provide separate delay and rate guarantees [36].

While the frameworks and mechanisms described until now are aimed at guaranteeing QoS even during overloads, traffic engineering [37] is the process of arranging how traffic flows through the network so that congestion caused by uneven network utilisation can be avoided. RFC 3272 [37] also discusses several management concepts such as online vs. offline optimisation, global vs. local
2.3. CHARACTERISTICS OF A QoS ARCHITECTURE

information, or centralised vs. distributed decision. Formulated as an optimisation problem, it means to minimise the maximum of link utilisation in the network. When all links are utilised to the same extent, the network tends to perform optimally in terms of packet losses, total delay, or bandwidth efficiency [1], given that no flow differentiation is sought. Constraint-based routing is an important tool for making the traffic engineering process automatic. The constraints are represented by QoS requirements like delay, bandwidth, cost, loss rate. To simplify the routing process and make it more suitable for traffic engineering, multi-protocol label switching (MPLS) [38] can be used. In the context of a DiffServ architecture, a centralised management entity called the Bandwidth Broker (BB) is proposed for admission control and resource allocation (router configuration) inside a network domain [39].

The above survey can be concluded with the following observation. While in the network community the notion of QoS is mostly associated with QoS provisioning as requested by the users, in the real-time community it is mostly associated with adaptive systems, where applications can be executed at different QoS levels in order to improve the performance of the whole system. We will come back to these alternative viewpoints when describing our work in the context of mobile networks.

2.3 Characteristics of a QoS architecture

The basic resource allocation problem is how to allocate the limited resources in a system such that client needs are addressed in the best possible way; in other words, such that resources are utilised in an optimal way and the quality of service agreed with the client is respected.

While in this thesis we do not propose a full-blown QoS architecture, most of the requirements are implicitly or explicitly assumed in our resource allocation schemes. Aurrecoechea et al. identify several characteristics of a QoS architecture [3]. We present here a more compact and slightly modified version. A QoS archi-
tecture can be divided into two essential parts: QoS specification and QoS enforcement.

2.3.1 QoS specification

QoS specification is concerned with acquiring the application level QoS requirements, which might also include some management policies. QoS specification parameters are usually different at the different layers of the distributed system. The application should be able to specify its requirements using high-level parameters e.g. picture quality, window size, for video media. These have to be mapped by the QoS system into lower layer parameters, e.g. throughput and delay. Besides performance-related parameters the specification could include other non-functional requirements too (fault-tolerance, availability, security). QoS specification may cover the following attributes:

- Quantitative performance specification. Flow\(^1\) specification parameters usually imply delay, throughput rates, jitter and loss guarantees.

- Qualitative specification (level of service). These specify the degree of commitment to the quantitative measures. They could be divided in the following classes: deterministic, stochastic, and best-effort. Usually, deterministic and stochastic guarantees require resource reservation (either for the worst-case or for the average case). Best effort requires no performance guarantee and implies scheduling with lowest priority. Relative qualitative guarantees might also be specified (e.g. always scheduled first, or twice as much allocated bandwidth).

- Cost of service. This specifies the price the user will pay for the service level. Without this factor there would be no

---

\(^1\)In the context of a generalised QoS architecture the flow refers to the end-to-end process of a distributed application, which implies the executions on nodes and transport over the network.
2.3. CHARACTERISTICS OF A QOS ARCHITECTURE

reason to choose lower levels of QoS. This is also a lever for arranging a good mix of service levels in the system.

- QoS management policy (application adaptation specification). Specifies application tolerance to QoS modifications, and actions to be taken if the QoS contract is violated. A QoS contract specifies the QoS level the system has committed to the user. It is usually negotiated when a new flow arrives into the system and depends on resources available and cost of service.

2.3.2 QoS enforcement mechanisms

Depending on the QoS requested by the user, the available resources in the system, and the optimisation criteria of the QoS management scheme, several mechanisms are possible to employ, as described below. QoS provisioning mechanisms are used at flow establishment and renegotiation phases. QoS control mechanisms refer to the path level QoS control mechanisms, and operate on a fast time-scale. Overall QoS management mechanisms operate at a system-wide level and on a slow time-scale.

QoS provisioning mechanisms

- QoS mapping. This is responsible for transforming higher level QoS requirements into lower level parameters, understandable by the respective layer. For example size and picture quality of a streaming video are mapped into throughput and delay requirements at transport layer that in turn generate a certain policy for the packet scheduler.

- Admission control. This is required to balance new resource requirements arising in the system compared to available resources. The decision depends on the system-wide QoS management policy. If available resources are reserved, and the end-to-end path reservation is successful, they are allocated.
• Resource reservation protocols. These protocols reserve resources in end-systems and network, and maintain the reservations as long as necessary.

QoS control mechanisms

• Flow scheduling. This regulates the flow’s access to different resources in the path (being CPU or network wire).

• Flow shaping. This regulates the flows based on performance specifications (by inserting delays). It allows the QoS architecture to commit enough end-to-end resources.

• Flow policing. This observes if the flow is behaving according to its specification and may constrain it into the specified shape.

• Flow control. Closed or open loops schemes may be used for controlling the flow, depending on whether the application is adaptive to fluctuations in the available resources or not.

QoS management mechanisms

• QoS monitoring. This allows the system to track the ongoing QoS level achieved by each layer.

• QoS maintenance. This compares the monitored QoS against the expected performance and takes the necessary adjustments in resource allocation to sustain the delivered QoS.

• QoS availability. This announces interesting changes in the provided QoS level to the application. Of particular interest is QoS degradation; if the system cannot maintain the QoS it has to notify the user, who chooses to adapt or to renegotiate (graceful degradation is strived for).

In the perspective of the above presented architecture, regarding QoS enforcement, we address higher level QoS management policies and mechanisms. Low level enforcement mechanisms, like
policers or schedulers are not in the scope of this work, but we assume them in place, to force ill-behaved flows to follow their specification.

Regarding QoS specification, we propose the usage of resource-utility functions that are semantically rich enough to represent a) the QoS levels, b) the user’s willingness to pay, and c) the negotiation potential of an application. Utility functions are a core ingredient of our algorithm, and they are the focus of the next chapter.
Chapter 3

Utility model

In this chapter we focus on the notion of utility functions. We first discuss two important topics for our resource allocation schemes: how to model application adaptation, and what type of overall system utility maximisation can be employed. Then we present the chosen utility model, that includes the relationship between resources, QoS levels, and perceived utility. Finally we present the instantiation of some utility functions, more specifically the utility functions we have associated with the traffic types used in our test scenarios.

3.1 Application adaptation

Different types of application might be accommodated by a QoS aware architecture. There might be applications with rigid requirements, which are either accepted or rejected, or adaptive applications able to function at different levels of quality, depending on what the system can offer. Many of the future generation mobile applications support different QoS levels. For example, multimedia services, either interactive or streaming, can decrease audio or picture quality to meet some bandwidth or delay restrictions, while applications like e-mail or file sharing can usually adapt to anything available.
This adaptation process is mentioned in the general QoS architecture characteristics in the previous chapter, but only as an interactive process. At admission control a QoS contract is negotiated and the system has to provide the agreed QoS. If the system can no longer sustain it, the user\(^1\) is signalled and has the choice of accepting the new conditions or renegotiating the QoS contract.

While the online negotiation mode is very flexible and gives a fine-grained control to the application, there are two major disadvantages with this method. First, there is a lot of signalling going on during the negotiation and renegotiation phases. Second, it is very hard to provide a resource allocation optimisation technique. For instance, in the admission control phase, a new application is requesting a certain amount of resources, currently not available. The system could either reject it or degrade some other application(s). But there is no knowledge on the adaptability potential of the other applications unless other renegotiation phases begin.

To eliminate these two problems, each application could specify its own adaptability to different service levels with the help of utility functions. These utility functions represent the benefit the application gets from different service levels. Furthermore, the QoS mapping mechanism should be able to construct resource-dependent utility functions. Then, the resource allocation algorithm could use these resource-related utility functions to provide an optimal, utility dependent allocation. Also, a direct mapping between utility and the amount a user is ready to pay for the service can be made, and can be useful when devising optimisation strategies.

If the adaptability of the application is specified as a utility function, it does not necessarily mean that a QoS contract cannot be signed between the provider and user. A major contribution of this thesis is guidelines on how such a contract can be specified, and detailing the consequences of breaking it.

\(^1\)A requested (or accepted) QoS level always depends on both the application and the user running it. While QoS is ultimately requested by the user, the type of application is determining the choices.
3.2 Utility optimisation and fairness

We mentioned that a quantitative measure of the resource dependent utility enables optimisation comparisons between the different users (applications) competing for system resources. The whole system’s performance can be specified as a function of the utility perceived by each partaker, according to the purpose of the system. The function describing the system-wide utility is similar to the welfare functions studied in economic sciences.

We present here the two most common welfare functions \([40]\). The system performance is denoted with \(W(u_1, u_2, \ldots)\), where \(u_i\) are the utilities generated by the participating applications. The min-max criterion which corresponds to the Rawlsian welfare function is \(W(u_1, u_2, \ldots) = \min(u_1, u_2, \ldots)\). Thus if we want to maximise \(W\) we have to distribute resources such that the utility of all participants should be equal. The second approach is the sum-of-utilities criterion and corresponds to the utilitarian welfare function \(W = \text{sum}(u_1, u_2, \ldots)\). As a consequence of maximising \(W\) resources are allocated according to efficiency (i.e. utility/resource ratio). A useful extension for both techniques is the possibility to specify relative priorities by multiplying the utilities with different weights. For instance in a mobile communication network this can be used for specifying different categories of users like emergency, gold, silver, and so on.

Both systems have their advantages and disadvantages. The sum-of-utilities appears quite suited for commercial systems where utility corresponds to revenue, even if some users might be strongly deprived of resources during overloads. The min-max criterion is better used in closed environments and where it is important that all applications run at the same utility level.

A straightforward example of the utilitarian criterion is an absolute priority-based scheduling policy, in which the higher utilities will always be preferred to lower ones. A straightforward implementation of the min-max policy is a share-based scheduler where the rates are allocated such that utility is equal for all participants.
3.3 Application utility

In our work we consider a user-centric utility view. We assume that utility is not an internal system parameter used for enforcing the system’s internal policies, but is directly specified by the user. We consider it to represent the satisfaction (benefit) that the user assigns to the results of running the application at a certain QoS level. Note that in the literature, references to QoS can point both to (a) the QoS that the infrastructure can offer to the contending applications and services, and (b) to the quality of the service or application perceived by the end user.Basically they two QoS types are closely connected. A better QoS provided by the infrastructure (e.g. bandwidth, delay, error ratio) translates to better QoS as perceived by the end user. In this work we refer to the latter, since it is the one that can be quantified with regard to its utility to the end user.

Now, it can be hard to assess the utility of a certain QoS level of an application, due to the fact that a certain QoS level may be valued different by different users. Moreover their valuation might change for different circumstances.

While it might seem difficult to put a value on the different combinations of QoS parameters, decision making theory (e.g. Analytical Hierarchy Process (AHP) [41]) and in particular, utility theory (e.g. Multi-Attribute Utility Theory (MAUT) [42]) has a strong tradition in e.g. economics, or medicine. For multimedia in particular there are subjective assessment methodologies (SAM) [43] from the International Telecommunication Union (ITU) and European Broadcasting Union (EBU), for evaluating different QoS set points. In the case of video for example, a QoS point represents a codec choice together with a specific bitrate, resolution, frame rate, etc. A general evaluation of video codecs [44] is published by the EBU. Obviously, individual preferences might vary.

Looking at the previous methodologies, the construction of the utility functions appears quite complex, and seen from the average user perspective it can appear too complicated. The previous
methodologies however, need not to be employed. For commercial systems the utility of a certain service level will most likely be linked to the price the user is ready to pay for the service. So for instance, if a fixed rate is preferred, a simple, one-step utility function can be used. To summarise, the utility approach does not require the usage of complicated evaluation techniques. It merely provides the possibility, and leaves the final decision to the user of the system.

As presented in Section 3.1, using utility functions eliminates the need for online QoS negotiations. A utility function tells the system all the acceptable resource configurations, and how they are valued. Thus the system can optimise resource allocation, i.e. it can maximise the overall system utility.

3.4 Q-RAM utility model

A seminal work in this area is the work by Lee et al [21, 20] in the framework of the Q-RAM project [45]. The Q-RAM project defines a comprehensive utility model that we use also in this work. The model can be summarised as follows: For every task a QoS set point is defined across several QoS dimensions, for example a video application can be defined by the following QoS dimensions: {framerate, resolution, video codec, audio sampling rate, end-to-end delay, etc.}. Every QoS dimension is composed by a set of feasible and relevant dimensional set-points. For example, the video codec can be one of \{cellb, nv, h.261\} or the framerate one of \{10, 20, 30\}. Some of the QoS dimensions are inherently discrete (i.e. video codec, resolution), others are quantised in order to show only relevant changes in the QoS (and to keep complexity low).

Thus, the quality space for a task $\tau_i$ is defined as follows $Q_i = Q_{i1} \times Q_{i2} \times ... \times Q_{iN_i}$ where $Q_{ij}$ is the $j^{th}$ QoS dimension and $N_i$ is the number of QoS dimensions for task $i$. Even if there are tasks that belong to the same application type, users of the system can

---

2 We have usually used the word application until now. Using task instead seems more appropriate, since a task is an instance of an application. The utility functions of two tasks can be different due to different users.
value the QoS differently, so for every task \( \tau_i \) there is a utility function defined on the set of quality points, \( u_i^q : Q_i \rightarrow \mathbb{R}^* \), where \( \mathbb{R}^* \) is the set of non-negative real numbers.

The QoS set points can be achieved only if enough resources are allocated to the application. The relationship between allocated resources and the QoS set points is not a function since, a) an application can choose between two or more algorithms that achieve the same quality but use different resources, and b) a certain resource allocation can be used by the application to improve the QoS along different QoS dimensions.

Thus let \( R_j \) denote the set of non-negative values representing the possible allocation choices of the \( j^{th} \) shared resource. The set of possible resource allocations, denoted as \( R \), is given by \( R = R_1 \times \ldots \times R_m \). Then a resource allocation choice \( r \in R \) and quality point \( q \in Q_i \) are in relation, \( r \models_i q \) (read as \( r \) satisfies quality \( q \)), if task \( \tau_i \) can achieve quality \( q \) using resource \( r \).

So, what is the value of a possible resource allocation for a given task? Combining the previous function and relation, resource dependent utility functions can be constructed. For a given resource allocation, different algorithms/configurations can result in different QoS set points for a certain task. Intuitively, the system should choose the QoS set point that gives the highest utility. Thus we can bridge over the QoS set points and define the following utility function \( u_i : R \rightarrow \mathbb{R}^* \), and \( u_i(r) = \max \{ u_i(q) \mid r \models_i q \} \).

Finally, we can define the overall system utility function \( u : R^n \rightarrow \mathbb{R}^* \) as the sum of the utility functions for all the tasks in the system, \( u(r_1, \ldots, r_n) = \sum_i u_i(r_i) \). This definition of system utility corresponds to a utilitarian welfare, and is suitable also for systems where profit is modelled as utility. A brief comparison to a min-max welfare system is made in section 3.2.

The optimal resource allocation problem

The optimal resource allocation problem is to find the values for \( r_i \) such that \( u(r_1, \ldots, r_n) \) is maximised. Let \( i \in [1, \ldots, n] \) index the tasks and \( j \in [1, \ldots, m] \) index the resources, thus \( r_i = \{r_{i1}, r_{i2}, \ldots, r_{im}\} \). Let \( r_{j_{\text{max}}} \) denote the maximum available amount
of resource \( j \). Thus we have the following formulation:

\[
\text{Maximise } u(r_1, ..., r_n) = \sum_{i=1}^{n} u_i(r_i) \quad (3.1)
\]

\[
\text{subject to: } \sum_{i=1}^{n} r_{ij} \leq r_{ij}^{\text{max}} \quad (3.2)
\]

\[
r_{ij} \geq 0 \quad (3.3)
\]

The previous resource allocation problem is NP-hard even in the case of single resource allocation [21]. Nevertheless, the single resource allocation problem is less complex, and efficient approximation algorithms, where results are with a known bounded distance from the optimal solution, have been proposed [21, 20]. We use an algorithm from Lee et al.’s single resource allocation as a base for our resource (re)allocation scheme, so we give a more detailed description of it in the next section.

Regarding the optimisation of the multi-resource allocation problem, works in the framework of Q-RAM [21, 46] propose a heuristic based on the notion of compound resource (an artificial resource built as a combination of all resources). This algorithm however, provides no bound on how far the results are from the optimal solution. For an optimal allocation, Lee [21] proposes a pseudo-polynomial solution based on dynamic programming, that actually grows exponentially with the number of resources, and would be infeasible for most systems.

Other works are concerned with a more specific case of multi-resource allocation. An example is bandwidth in a multi-hop network environment [47, 48]. In this case, there are several pools of the same type of resource (the bandwidth associated with the links/nodes in the network), where end-to-end connections use different subsets of these bandwidth pools. A novel algorithm for this case is proposed in Chapter 6.
3.5 Optimising single resource allocation

We assume that each task has a resource-utility (R-U) function\(^3\), which is specified by the user of the task, \(u_i : \mathbb{R}^* \rightarrow \mathbb{R}^*\), \(i\) identifies the task, and \(u_i(r)\) describes the utility that accrues given a resource level \(r\). In our work, the resource represents bandwidth available in a wireless network, and the tasks are application-level connections contending for the bandwidth. As a reflection of the variety of applications, utility functions may exhibit different patterns: concave, convex, linear or step functions. The only obvious restriction is that a R-U function should be non-decreasing.

For the ease of utility specification, and for keeping implementation complexity low, it is necessary to quantise utility functions using a small set of parameters. Figure 3.1 presents a Utility function before and after quantisation, such that the utility function can be represented by a list of quantisation points, in increasing

---

\(^3\)For the rest of the thesis the only type of utility functions used are R-U functions, and we will use these terms interchangeably.
order of resource consumption:

\[ u_i = \left( \left( \frac{u_{i,1}}{r_{i,1}} \right), \ldots, \left( \frac{u_{i,s}}{r_{i,s}} \right) \right) \]

where \( s \) is the number of utility levels of task \( i \), and \( u_i(r_i) = u_{i,k} \) is the accrued utility for an allocated resource \( r_i \in [r_{i,k}, r_{i,k+1}) \), where \( 1 \leq k \leq s \).

Utility maximisation.

Next we describe the resource allocation problem. We assume that the utility of a system is the sum of the utilities of all tasks in the system. Then, the utility maximisation problem can be formulated as follows:

\[
\text{Maximise } u(r_1, \ldots, r_n) = \sum_{i=1}^{n} u_i(r_i) \quad (3.4)
\]

subject to:

\[
\sum_{i=1}^{n} r_i \leq r^{\text{max}} \quad (3.5)
\]

\[
r_i \geq 0 \quad (3.6)
\]

where \( u : \mathbb{R}^n \rightarrow \mathbb{R}^* \) is the system-wide utility, \( r_i \) are the allocation variables to be solved, and \( r^{\text{max}} \) is the total available resource.

The 0-1 knapsack problem can be directly reduced to the above utility maximisation problem [21], which makes the latter NP-hard. Nevertheless Lee et al. present several approximation algorithms to solve it. In what follows, we present the algorithm that we use as a basic ingredient in our bandwidth allocation scheme from Chapter 4.

The convex hull approximation algorithm

The \texttt{convex_hull_opt} algorithm is a low complexity algorithm that nevertheless performed within 99% of the optimal solution in the experiments by Lee et al. [21, 20]. Figure 3.2 presents an abstract
version of the original algorithm (originally referred as asrmd1 [20]).

As a first step, the algorithm approximates all utility functions by their convex hull frontiers, which are piece-wise linear, concave functions. Next, all convex hulls are split in segments corresponding to their linear parts. Note that a segment’s “resource need” is the resource increment between two allocation levels. Next, all the segments are ordered by decreasing slope order (see Figure 3.3), and resource is allocated in this order until depleted.

Thus, an allocation to a task equals to the sum of the resource amounts of its “allocated segments”. The concave form of the convex hull ensures a consistent allocation. Note that the slope of segment \( j \), \( (u_{i,j} - u_{i,j-1})/(r_{i,j} - r_{i,j-1}) \), represents its efficiency in terms of contribution to the system utility.

The complexity of the algorithm is \( O(nL \log n) \), where \( n \) is the number tasks, and \( L \) is the maximum number of utility levels of a utility function.
For the theoretical worst case difference between the optimum allocation and the convex hull optimisation algorithm the following lemma is given by Lee [21]. Let $\delta_i$ denote the maximum difference between the convex hull of a utility function and the utility function itself, for any of the tasks involved. For instance in Figure 3.4 the maximum difference between the convex hull (the thick dashed line) and the utility function (the continuous line) is just before the 0.5 bandwidth point. Consider now a set of tasks, with utility functions $u_i$, contending for a maximum available resource amount $r^{max}$. Let $\xi$ be the highest $\delta_i$ of all the utility functions in this set of tasks, $\xi = \max_{i=1}^{n} \delta_i$. Let $U$ be the system-wide utility result obtained by convex_hull_opt on this set of tasks. Let $U_{opt}$ be the optimal utility obtainable for this problem instance.

**Lemma 3.5.1** $U_{opt} - \xi \leq U \leq U_{opt}$. 
Proof The convex_hull_opt algorithm provides a feasible allocation, thus $U \leq U_{opt}$. Now, let’s change the algorithm in two places, first we eliminate the condition in line 10, allowing fractional allocations. We also compute a different utility instead of line 16, $U_{aug} = \sum_i u_i'(r_i)$, calculated using the convex hull frontiers that replaces the original functions. By this, we have transformed the allocation problem to a problem equivalent to the fractional knapsack problem (with the linear segments of the convex hull frontiers playing the role of items). Moreover, the fractional knapsack problem is optimally solved by the modified greedy algorithm, so $U_{aug}$ is an optimal solution to the modified problem. Since the convex hull frontier is at least equal to the original utility function for any allocation, $U_{opt} \leq U_{aug}$. Finally, we observe that $U_{aug} - U \leq \xi$, since the two allocations differ only in a potential partial allocation for a segment of a utility function. Combining the last two inequalities we arrive at $U_{opt} - \xi \leq U$.

For an example of the difference between $U_{aug}$ and $U$ see the “optimal utility” graph in Figure 3.3.

3.6 Utility setup for network traffic

Having described the general approach of representing utility functions and allocating resources, we now go on to instantiate the model. We use bandwidth as a resource, with different types of connections contending for it. In order to evaluate the algorithms presented in this thesis, we adopt a traffic mix that we regard as representative for communication networks. The application mix was previously used by Oliveira et. al [49] and in the Rate Based Borrowing Scheme (RBBS) [2]. As the RBBS is not using utilities, we augmented the model by adding utility and associating each of the six application groups with a utility function. Table 3.1 summarises the characteristics of the traffic mix.

Column 2 of Table 3.1 presents the maximum bandwidth requested by the connection. To simulate connections of different sizes, the value of this parameter is not fixed, but follows a
Table 3.1: A traffic mix setup

<table>
<thead>
<tr>
<th>Applic. Group</th>
<th>Bandwidth Requirement (Kbps)</th>
<th>Connection Duration (sec)</th>
<th>Examples</th>
<th>RBBS class</th>
<th>TARA class</th>
<th>Utility scaling factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>max</td>
<td>avg</td>
<td>min</td>
<td>max</td>
<td>avg</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>60</td>
<td>600</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>Voice Service &amp; Audio Phone</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>60</td>
<td>1800</td>
<td>300</td>
</tr>
<tr>
<td></td>
<td>Video-phone &amp; Video-conference</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>6000</td>
<td>3000</td>
<td>300</td>
<td>18000</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td>Interact. Multimedia &amp; Video on Demand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>120</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>E-Mail, Paging, &amp; Fax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>512</td>
<td>256</td>
<td>30</td>
<td>36000</td>
<td>180</td>
</tr>
<tr>
<td></td>
<td>Remote Login &amp; Data on Demand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1000</td>
<td>10000</td>
<td>5000</td>
<td>30</td>
<td>1200</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>File Transfer &amp; Retrieval Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

geometric distribution with the given minimum, maximum, and mean, as shown in the respective subcolumns. Note that applications are adaptable, so the actual allocated bandwidth may be lower than the requested bandwidth, but not higher, since the requested bandwidth is associated with maximum needs. Column 3 presents the duration of the connection, that again is a stochastic variable following a geometric distribution with the given minimum, maximum, and mean. Column 4 identifies the application types. Column 5 presents the RBBS class, and in column 6 we present the TARA class, a main parameter particular to our allocation schemes. It describes the flexibility of a certain class of applications to resource reallocations and allows us to differentiate connections that require stronger resource guarantees compared to more flexible (best effort) connections. Both class parameters will be covered in Chapter 4.

Finally, Column 7 (the rightmost column in Table 3.1) gives the relative importance between application groups. It represents the utility/bit associated with the maximum requested bandwidth. For example, one might be ready to pay roughly three times more for a video-phone conversation than for an audio conversation. The video conversation has a bandwidth demand of 256 kbps, while the audio-phone requests only 30 kbps. Thus the utility per bit for the audio-phone should be almost three times higher than for the video-phone. Consider row 3 in the table, if the reques-
ted bandwidth is 6,000 kbps then the utility for this bandwidth is
6,000,000 \times \frac{1}{10} = 600,000. Having fixed the relative difference at
the maximum level, all the other utility values of the utility func-
tion are calculated following an appropriate shape of the function.

The shapes of the utility functions for the six application groups
are presented next. On the x-axis we plot the relative bandwidth
with respect to the maximum requested by the application (de-
scribed in Column 2 of Table 3.1). On the y-axis we have the
relative utility with respect to the maximum utility, which is ob-
tained by multiplying the maximum requested bandwidth with the
utility scaling factor in Column 7 of Table 3.1. Assigning utility
values is always a subjective problem, so we chose some common
sense values and also consulted with Ruben et al. [50] who per-
formed a study at Ericsson Cyberlab in Singapore and had access
to current conceivable business models. In the next figures the
continuous line corresponds to the quantised utility function (the
one that defines the “contract” between user and provider), the
dashed line is the convex hull frontier, used by the allocation al-
gorithm, while the dotted line is just underlining the shape of the
utility function.

For the audio phone we used a classical sigmoid shape, as
presented in Figure 3.4. Sigmoid utility functions are intuitive and
expected for more advanced services requiring a certain timeliness
(such as audio and video conferencing). It is assumed that in the
beginning, the amount of resource is “not enough” for an “accept-
able” quality, so the utility function slowly grows until it hits a
sweet spot, when the service becomes acceptable and the quality
quickly raises. Finally, the quality is “good enough” and percept-
ible changes can be gained only by using big amounts of resources.

The next two utility functions concern video connections. For
the first one, which does not use a big amount of resources and
where less video quality should be acceptable we used a concave
utility function, such as in Figure 3.5. For the second one, we
considered that a second utility bump cannot be achieved without
3.6. UTILITY SETUP FOR NETWORK TRAFFIC

![Utility function shape for application group 1](image1)

**Figure 3.4:** Utility function shape for application group 1

![Utility function shape for application group 2](image2)

**Figure 3.5:** Utility function shape for application group 2

the use of more costly algorithms (e.g. for quality fast-motion scenes) so we used the function presented in Figure 3.6.
Application groups 4 and 6 have been considered “best effort” connections that can adapt linearly to any resource allocation so we have a linear utility function, as in Figure 3.7. Note that even if they have the same function shape their utility/bit (i.e. overall importance) is very different (i.e. 3 to 1/7, see Table 3.1). Finally, for the interactive application group 5 we have considered a sweet spot around 200 kbps as acceptable transmission rate, so the shape in Figure 3.8.

Unless otherwise stated, the above traffic characteristics apply to all the experiments throughout the thesis. Other traffic characteristics (such as arrival rate), network topology and mobility are presented in the respective chapters.

With all preliminaries presented, in the next chapter we introduce the Time-Aware Resource Allocation scheme (TARA) where we model and include the effects of resource reallocation into a global allocation optimisation scheme.
Figure 3.7: Utility function shape for application group 4 and 6

Figure 3.8: Utility function shape for application group 5
Chapter 4

Adaptive (re)allocation

In this chapter we study the problem of providing optimised allocation in highly dynamic systems. More specifically we propose a time-aware admission control and bandwidth allocation scheme in the context of a future generation cellular network. The quality levels of the connections are specified using discrete resource-utility functions, and bandwidth allocation aims to maximise the aggregated utility of the system. The key observation is that the available resources might quickly change and thus periodic reallocations are needed to keep the performance optimised. Reallocations however, can have a disruptive effect on some applications, and we identify this information, classify it, and use it in the allocation decision process. The evaluation of our approach shows a superior performance compared to a recent adaptive bandwidth allocation scheme, the Rate Based Borrowing Scheme (RBBS). In addition we have studied the overhead imposed on the infrastructure by a reallocation round, and we present an algorithm that efficiently reduces the number of reallocations, while maintaining utility within a given bound.

4.1 Overview

As mentioned in section 2.2, link resource allocation has been extensively studied in the Internet community. Several resource
reservation architectures together with enforcement mechanisms have been proposed for guaranteeing QoS to the flows in the network.

However, a cellular communication network is different from the Internet, which makes the QoS allocation problem more complex. Due to the nature of the wireless link, resource availability is both more uncertain and more acute. Theoretically (assuming it is feasible to implement), a resource reservation protocol in the Internet can guarantee the decided QoS for the accepted flows. This is not true anymore for wireless networks. First, due to mobility, a user might move to a cell where there is no bandwidth available, after being admitted in another cell with no resource shortage. Secondly, the effective bandwidth of the wireless link may fluctuate due to fading and interferences. Thus, a bandwidth decrease might deprive users from resources even after they have been admitted in the system.

Moreover, the wireless spectrum available for communication is fixed and limited. Compared to the Internet, resource availability cannot be solved by over-provisioning. While this increases the need for optimised resource allocation algorithms, it also allows us to make the following simplification. Since the capacity of the wire-line links can be easily augmented, we can assume that the wireless link is the bottleneck resource, and that the wire-line part of the network never gets overloaded. Therefore, the end-to-end QoS perceived by a connection\(^1\) depends only on the allocation decisions on the wireless link.

The above factors describe a system where bandwidth availability is highly variable in time, and the system may often find itself in an overload situation. For the bandwidth manager to take the best allocation decisions, we assume that a quantitative measure of the utility (benefit) generated by each connection is available in form of bandwidth-dependent utility functions. System utility is

\(^1\)By connection we refer to an end-to-end application-generated connection (e.g. TCP connection) and not to the radio connection between MS and BTS. Since each connection represents an application, the two terms are regarded as interchangeable in the following text.
calculated as the sum of the utility generated by each connection (and can be easily linked to network operator revenues). Thus, optimal allocation maximises system utility.

As already mentioned, resource reallocation might be needed in order to improve total utility (if bandwidth becomes available, i.e. a connection finishes or leaves the cell) or to provide graceful degradation (when bandwidth has to be reallocated to new connections or incoming handovers). In order to make more informed decisions on resource reallocation, in addition to utility functions, we also consider the fact that different applications react differently to resource reallocation. For example, if a hard real-time application is degraded, we would expect no utility from this application, and the resources invested so far would be wasted. On the other hand, an FTP session has no restriction to switch between different resource allocation levels, no matter how often.

Therefore, we propose TARA, a Time-Aware Resource Allocation scheme that aims to provide bandwidth allocation/reallocation based on the utility-efficiency (utility per bandwidth) of the competing connections. The novelty is that our scheme identifies how resource reallocation decisions affect the utility of the application, and integrates this information into the bandwidth management algorithm. Based on their flexibility to reallocations, we have categorised applications in three classes: non-flexible, semi-flexible and fully flexible. The time at which a reallocation decision is taken is also very important. Because of the invested resources, disconnecting a connection when it is nearly finished creates a larger utility loss than if it is dropped just after start and bandwidth has been invested for a small period of time. Thus, the system has to be aware of the age of the connections to take a good (re)allocation decision. In addition to this, two more factors have been considered when reallocating. First, the drop penalty allows the user to specify its dissatisfaction of being first accepted and then rejected. Second, the sensitivity of some of the connections with respect to the frequency of reallocations is considered.

To evaluate our scheme we have built a simulation platform in which we compare our approach with a baseline version that is un-
aware of the previously mentioned factors, and a recent published algorithm, the RBBS [2].

Finally we consider the overheads created by our bandwidth allocation scheme as a result of the periodic reallocation. Reallocations increase the resource demand from the infrastructure (e.g. CPU time for executing associated control functions or additional bandwidth for signalling). Thus, by performing too many reallocations in order to improve utility, the system might get overloaded. Consequently, service availability will suffer, and the generated utility will decrease; contrary to what was intended. We present and evaluate a new algorithm for controlling this overhead.

4.2 Related work

Similar in application area, El Kadi et al. [2] and Oliveira et al. [49] propose adaptive bandwidth allocation schemes for wireless cellular networks, without an explicit use of utilities. Still, the schemes use a flexible allocation approach, where connections specify a mandatory minimal bandwidth and an ideal maximal bandwidth. There is also a coarse mechanism of differentiation, i.e. real-time connections are more important than best-effort, handovers more important than new connections. In the work of Oliveira et al. [49], the allocated amount of bandwidth during the stay in a cell is fixed, and can change only at a handover. El-Kadi et al. [2] provide a more adaptive scheme, by allowing bandwidth to be borrowed from already accepted connections. Although the scheme is adaptive, it does not include a quantitative measure of the importance of the different connections.

While many applications can be run at different QoS levels, we consider that without a quantitative notion of importance, the QoS management system will not have the ability to efficiently prioritise allocations during overloads. To this end, utility functions can be used to specify a quantitative measure of the QoS perceived by the application. Chen Lee et al. [20] use resource-utility functions in a QoS management framework with the goal to maximise the total utility of the system. They propose two ap-
proximation algorithms, and compare the run-times and solution quality with an optimal solution based on dynamic programming. In our work we build on top of such an utility maximisation algorithm (presented in Section 3.5), but we also take into account bandwidth reallocations and their effect on the connections’ generated utility.

Rui-Feng Liao et al. [51] use “utility functions” in a bandwidth allocation scheme for wireless packet networks. However as opposed to maximising the total utility of the system, they provide “utility fair allocation” to the connections. Their algorithm extends “max-min fair allocation”, with utility replacing bandwidth as the fairness criterion. While this scheme provides equality to all connections, it might have counterproductive effects during overload conditions, since it degrades all the existing connection to a low common utility.

Abdelzaher et al. [52] propose a QoS-adaptive Resource Management System for Internet servers. A QoS contract is used to specify acceptable QoS levels, along with their utility. There is no restriction in reallocations (similar to our fully-flexible class). However, there is a “minimum” allocation level that must be guaranteed. Otherwise a “QoS violation” penalty (similar to our drop penalty) is incurred. They compare an optimal allocation policy based on dynamic programming with a first-come first-serve policy where resources are not reallocated.

A system that also addresses resource allocation in mobile networks is the “TIMELY Architecture” proposed by Bharghavan et al. [53]. While we are concerned with the allocation at a “policy level” their system coordinates allocation from the MAC-layer, through resource reservation and resource adaptation to the transport layer. Moreover, an end-to-end allocation over both wireline and wireless links is attempted. They employ a revenue model with a 4-tuple: revenue function, termination credit (similar to our drop penalty), adaptation credit (similar in function to what we call adaptation time) and an admission fee. Maximising the revenue (based on the max-min criterion) is one of the criteria used during allocation and adaptation. On the other hand, the
same 4-tuple is used for all flows. While simplifying allocation, it prevents differentiation (as different importance or different assurance needs) between flows. In our work we assume that the QoS specification (as R-U functions, flexibility classes) is connection specific, and during allocation the system uses these parameters to differentiate between connections.

An optimal sampling frequency assignment for real-time wireless sensor networks is proposed by Liu et al. [54]. In their model, the underlying network uses dedicated channels to communicate between neighbours such that interferences are avoided. Nevertheless, a flow traverses several channels so the bandwidth allocation of the different wireless channels is not independent (as opposed to our work). A utility loss index, which depends on the sampling frequency and has a convex form, characterises QoS and must be minimised across the network in order to optimise the system. Two algorithms are proposed, a centralised that is better suited to small networks and a distributed one that converges in several iterations, and works better for large networks.

In a more classical real-time approach, Richardson et al. work at packet scheduling level in their QoS provisioning system [55]. When a packet is in danger to be late, its scheduling priority is calculated based on the value of the connection it belongs to, otherwise EDF priority is used. The algorithms provides an non complicated, value-based scheduling that works better than EDF during overloads. However if can be inefficient (i.e. packet size is not considered, only value), and congestions are only mitigated, not prevented. By working on a higher level, we are able to take into consideration the characteristics of the connections and can optimise the allocation a priori, such that congestions are avoided.

4.3 Application area

In this section we give a short presentation of a 3G cellular network, chosen as an application area for our work. While the information presented in this section is not essential, it will however
4.3. APPLICATION AREA

present the reader with basic notions such as base station, handover, etc.

We base our descriptions on the Universal Mobile Telecommunications System (UMTS), a third generation (3G) mobile radio system [56, 57, 58]. Functionally UMTS is divided into three parts: the UMTS Terrestrial Radio Access Network (UTRAN), the UMTS Core Network (CN), and the Mobile Station(s) (MS). The MS connects to UTRAN wirelessly through the air interface. WCDMA (Wideband Code Division Multiple Access) is used as wireless transmission technology and enables communication speeds up to 2 Mbps. The UTRAN part deals with radio-related issues while the CN should be seen as a means of collating various generations of telecommunication networks and Internet, and supporting both channel switched and packet switched services.

Figure 4.1: The UMTS Radio Access Network

The resources we consider for management are part of the UMTS Terrestrial Radio Access Network (UTRAN), shown in Figure 4.1. The radio access network consists of several Radio Network Subsystems (RNS). A RNS is composed of a Radio Net-
work Controller (RNC) and a set of Radio Base Stations\(^2\) (BS), which in turn serve one or several operating cells. The CN, RNC, and BS are connected through a high-speed infrastructure (e.g. Asynchronous Transfer Mode - ATM).

The BS is the physical unit for radio communication with the MS and its main task is to convert data to and from the radio interface. In addition to that, it measures the strength and quality of the transmission and sends it to the RNC.

A user typically connects to a BS within a cell, and moves around from cell to cell over time. When the user crosses into another cell, a handover takes place, that is, the communication with the initial BS is released and the new BS takes over the communication to the MS.

The entity responsible for managing the radio resources in the radio access network is the RNC. Thus, the RNC has to execute several control functions, which can be divided into two main groups: traffic control functions, that are generated by user activity, and operational and maintenance functions requested by higher network management layers. The traffic control functions are in their turn divided into connection management functions and mobility management functions.

Some of the most important traffic control functions the RNC is responsible for are:

- Signalling Connection Setup/Release. The MS establishes a signalling connection link, Signal RadioBearer (SRB), in order to transfer control signals between the RNC and the MS.
- Radio Access Bearer (RAB) Setup and Release. These provide the ability to establish and release connections for transferring user data.
- Channel Switching. Information is transported over the radio interface by means of different types of channels. A “dedicated” channel is used exclusively by a single MS, usually for higher bit rate transfers. For lower bit rates or bursty traffic, several MSs can share one of the “common” trans-

\(^2\)Node B in the 3rd Generation Partnership Project (3GPP) specifications
port channel types. Even when not transferring any data the MS’s are using a certain common channel to keep in touch with the network. Both dedicated and common channels can use a wide range of bitrates (between 15 and 1902 kbps). “Channel type switching” is used for switching between the different channel types, depending on how active the traffic is. Another function is “channel rate switching”, and is used to accommodate a dedicated channel to the bitrate demands of the user.

- Soft/Softer Handover. Soft handover is performed when a user moves between cells controlled by different BS. If the same BS controls both cells it is a softer handover.
- Admission Control. This is performed to avoid the overload of the wireless link. New connection establishment, handovers and channel switches are subject to admission control.
- Power Control. This regulates transmission power of signal between BS and MS. It has to ensure both good signal quality and low interference.
- Cell Update and Paging. Cell update is used to inform UTRAN about the new location of the MS when it changes a cell. Paging is used to inform the MS of an incoming connection.
- Congestion Control. If the wireless link is overloaded congestion control frees up some bandwidth by reducing the rate of, or by disconnecting existing connections.

Operation and Management (O&M) functions typically include service provisioning, fault management and recovery, configuration management, and functions for billing support. Note that our bandwidth allocation scheme addresses a general utility maximisation problem, and is not constrained to a 3G environment. That is, we approach the problem from a more abstract level and do not address how the allocated bandwidth can be mapped to different radio access bearers (RABs) or how channel switch functions can be used to perform reallocations.
The 3GPP [59] has an incipient definition of four general QoS classes, together with acceptable parameters for e.g. transmission rates, bit error rates, transfer delay, etc. The classes are broadly differentiated by their delay or loss requirements. Conversational and Streaming are intended for real-time flows. Applications for the conversational class include voice and teleconferencing services. Due to the conversational pattern this class is very sensitive to delay and delay variation. The streaming class represents one-way audio and video transmissions. Since the transmission is one-way, the delay requirements are not as stringent. Due to buffering possibilities the streaming class is also less sensitive to delay variations. Interactive is intended for services like web browsing or telnet, where much larger delays, compared to the previous classes, are acceptable. Background corresponds to delay insensitive services like SMS, E-mail or FTP. The latter two are considered data services and should provide better fault tolerance (using better channel coding and retransmission mechanisms), than the conversational or streaming classes.

We regard our work as complementary to these QoS specifications. Basically the 3GPP standard identifies some broad application types, and defines acceptable QoS levels along several QoS dimensions. In our work we do not try to define such QoS levels or to develop mechanisms to implement them. What we aim for is a global policy to optimise allocation for any traffic mix, assuming that we know the amount of needed resource and the returned utility.

### 4.4 Reallocation consequences

The utility functions presented in Chapter 3 describe the resource-utility dependency in a static manner. In a dynamic system, where resources need to be reallocated, the utility given by an R-U function represents only a momentary value \( u_i(t) \). A better measure of the utility generated by a connection would be its utility accumulated in time, i.e. the utility generated by the connection over its entire duration.
4.4. REALLOCATION CONSEQUENCES

If, for some application class, the accumulated utility of a connection \( u^a_i \) corresponds to the integral of all the momentary utilities, that is \( u^a_i = \int_0^T u_i(t) dt \), then the following equality holds:

\[
 u^a = \sum_{i=0}^n u^a_i = \sum_{i=0}^n \int_0^T u_i(t) dt = \int_0^T \sum_{i=0}^n u_i(t) dt = \int_0^T u(t) dt
\]

The system-wide utility accumulated over time is denoted by \( u^a \), and the momentary system-wide utility by \( u(t) \). \( T \) represents a time interval. The above equation shows that in this case, the maximisation of \( u^a_i \) can be achieved by maximising \( u_i(t) \) at each time point \( t \).

However, for many application classes \( u^a_i \neq \int_0^T u_i(t) dt \). For example, there are applications with strong needs for resource assurance. That is, if the initial agreed resource amount is degraded, then all the potential utility generated until that moment is lost. In the end \( u^a_i = 0 \), and resources allocated to it since its arrival have been wasted. Therefore, our allocation algorithm needs to take into account the effect that reallocations have on the accumulated utility of the connections.

The main advantage of using utility functions is that the user (application) has the possibility to quantitatively specify the value he is attaching to the results. Accepting a connection and allocating a certain amount of bandwidth is similar to negotiating and signing a QoS contract between the user and provider. Since the R-U functions provide a framework to specify all the acceptable levels, the negotiation phase (and the associated overheads) can be skipped. In the same line of thought, each reallocation would amount to a breach of contract and signing of a new contract. Because of different application types or user preferences, different connections have different tolerance to bandwidth reallocation. The challenge is to find a representative (and small) set of parameters that satisfactorily describe the reallocation effects for a large set of applications. The vision is to give the user the possibility to specify a value for these parameters so that the system can take the right (re)allocation decision.
We have thus identified three factors that affect the perceived accumulated utility of a connection: the flexibility (adaptability) to reallocations, the sensitivity to a complete disconnection as opposed to only a bandwidth reduction (drop penalty), and the sensitivity to the frequency of reallocations (adaptation time). The following subsections describes these in more detail.

4.4.1 Flexibility classes

We first divide the applications into three broad classes depending on their flexibility with respect to reallocations.

**Class I** represents non-flexible connections. They require strict resource assurance to fulfil their mission. That is, once accepted (with initial utility $u_i^{\text{init}}$), the resource amount cannot be renegotiated. If the management system cannot assure the initial resource amount at any time-point during the lifetime of the connection, there will be no utility gained for the whole duration of the connection, and already invested resources are wasted. If a class I new connection is accepted, any subsequent reduction of bandwidth is equivalent to dropping the connection. Since it uses the same amount of resources during its lifetime, increasing the bandwidth brings no benefit. If the connection is not dropped, the accumulated utility of the connection is calculated by this formula: $u_i^a = u_i^{\text{init}} \times \text{duration}$. Examples are hard real-time applications, critical control data, real-time data streams.

**Class II** represents semi-flexible connections. These are applications that are judged by their worst moment in their lifetime. For this type of connection, the lowest utility (respectively bandwidth) experienced during its lifetime is used for calculating the utility for the whole duration: $u_i^a = u_i^{\text{min}} \times \text{duration}$. Compared to class I, a resource degradation, while diminishing utility, is not disastrous. However, once a certain level reached, the results cannot be improved if the resource allocation is increased at a later point. For example, users often remember the worst portion of a multimedia stream, or a distributed game. Another good example is sensor readings where the resolution bound is important. The
4.4. REALLOCATION CONSEQUENCES

resolution of the whole stream is the lowest resolution from all the readings.

Class III represents fully-flexible connections. These are the connections with no real-time requirements, and they can adapt to both increases and decreases of the bandwidth. The accumulated utility is the sum of all the momentary utilities over the total duration: \( u^a_i = \int_0^{\text{duration}} u_i(t) \, dt \). Examples are fetching e-mail, file transfer, or any type of connection in the “best effort” category.

A real-word application could be a combination of different connections of different class. For example it could consist of two parts, a mandatory one that is class I and a fully flexible, class III. Note that the shape of the R-U function does not depend at all on the class of the connection. The class does not affect the initial allocation possibilities, but only describes the effects at subsequent reallocations.

4.4.2 Drop penalty

We assume that disconnecting (dropping) a connection before its natural end brings its accumulated utility to zero. This shows that invested resources are wasted by such a decision. Moreover, resources have been invested also on the user side, and are lost at disconnection. Therefore, the user should be able to specify a certain drop penalty, which represents the customer dissatisfaction when a connection is disconnected after being admitted into the system. Let \( P_i^{\text{drop}} \) be the penalty for dropping a certain ongoing connection. Thus, if disconnected, the accumulated utility of the connection becomes negative, \( u^a_i = -P_i^{\text{drop}} \). If utility is used in calculating the revenue of the network operator, a negative utility implies some form of compensation to the user.

4.4.3 Adaptation time

Flexible (class III) applications can adapt to both increases and decreases in bandwidth. In a dynamic environment, these connections might be subjected to very frequent reallocations. But even these flexible applications might need a certain amount of time
to adapt to the new “mode” after a reallocation. For example, some algorithms for encoding, encryption, compression could be changed, some computations need to be restarted. Performing frequent reallocations might be worse than keeping a connection at a constant, lower resource level. A specified adaptation time is a way to reflect the minimum time between reallocations, in order for a connection to gain the expected utility.

The effects of not respecting a “minimum adaptation time” could greatly differ for different connections. Nevertheless, we propose the following performance degradation model. Let $I_i$ represent the time between two bandwidth reallocations, and $A_i$ a specified adaptation time. Now, if $I_i < A_i$, then we assume that the utility generated during the $I_i$ interval is only $I_i/A_i$ of the utility that would be generated under normal circumstances. That is, the utility gained in this interval $(u_i \times I_i)$ is diminished with an adaptation penalty, $P_{i}^{\text{adapt}} = u_i \times I_i \times (A_i - I_i)/A_i$. Note that the shorter the $I$ the higher the penalty is (compared to the gained utility in the interval $I$). Conversely, if $I$ takes values closer to $A$ the penalty tends to zero. Classes I and II should not be subject to frequent reallocations (bandwidth increases are useless, decreases are few and bound by the allocation levels of the utility function). Thus, this penalty is meaningful only for class III connections.

### 4.5 Dynamic reallocation

Because of the highly dynamic environment, constant reallocation is needed in order to obtain the best results. Basically, whenever a new connection is established, or handed over, or a connection ends, a new reallocation might be needed to improve system utility.

In a large system, an event-based allocation policy may lead to an unacceptable high call rate to the (re)allocation algorithm. This overhead can be controlled by employing a periodic reallocation method. The new connections and handovers arriving in between two allocation rounds are put in a queue that will be processed at the beginning of the next allocation period. For reas-
onably low values of the reallocation period, the process should be transparent to the user.

In Section 3.5 we mentioned a near-optimal allocation algorithm that uses the R-U functions as input. The algorithm orders and accepts connections based on their efficiency. We keep this as a base allocation algorithm. However, since the original R-U functions do not describe the history of a connection, we have to take the additional factors that describe the effects of reallocations (described in Section 4.4) into account. Therefore we create “artificial” R-U functions by modifying the original R-U functions at every (re)allocation time point. For instance, in an ongoing class I connection, resources have been invested for some time. The corresponding potential utility however, will only be gained if the connection is not dropped. When compared to a new connection, the ongoing connection comes with this earned utility so far, that effectively increases the efficiency of the connection over the rest of its lifetime. Thus by modifying the R-U functions we make the connections of all ages and classes efficiency-wise comparable.

While at each allocation point we optimise based on the utility-efficiency of the different connections, only a clairvoyant algorithm that knows all the future arrivals could provide a truly optimal allocation. For instance a class I connection accepted at some point might be dropped if an increased number of higher efficiency connections arrive at a later point. By not accepting it in the first place the system would have avoided paying the drop penalty. As clairvoyance is not a realistic assumption, the only other possibility would be do predictions based on profiling past arrivals. Even this option is considered as unrealistic, and our scheme will try to optimise based solely on current conditions and previous allocations.

In the next subsections we explain how the modified R-U functions are constructed at each reallocation. Table 4.1 summarises the parameters utilised in the process.
Table 4.1: Notation summary

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_{i}$</td>
<td>the original R-U function of conn. $i$</td>
</tr>
<tr>
<td>$u_{i}^{age}$</td>
<td>the age-modified R-U function of conn. $i$</td>
</tr>
<tr>
<td>$u_{i}^{drop}$</td>
<td>the drop penalty modified R-U function of conn. $i$</td>
</tr>
<tr>
<td>$u_{i}^{adapt}$</td>
<td>the adaptation-time modified R-U function of conn. $i$</td>
</tr>
<tr>
<td>$b_{i}(l)$</td>
<td>the bandwidth of conn. $i$ corresponding to QoS level $l$</td>
</tr>
<tr>
<td>$t_{i}^{drop}$</td>
<td>the drop penalty of conn. $i$</td>
</tr>
<tr>
<td>$t_{i}^{max}$</td>
<td>the duration of conn. $i$</td>
</tr>
<tr>
<td>$t_{i}^{age}$</td>
<td>the current age of conn. $i$</td>
</tr>
<tr>
<td>$t_{i}$</td>
<td>the time passed since the last allocation for conn. $i$</td>
</tr>
<tr>
<td>$A_{i}$</td>
<td>the adaptation time of conn. $i$</td>
</tr>
<tr>
<td>$u_{i}^{a}$</td>
<td>the value of the time-accumulated utility for conn. $i$</td>
</tr>
<tr>
<td>$u^{a}$</td>
<td>the value of the system-wide time-accumulated utility</td>
</tr>
</tbody>
</table>

### 4.5.1 Age and class influence

To get a feeling for why age modifications are needed, we start by giving an example of a reallocation decision where the original, unmodified R-U function is used. Assume there is a class I or II connection $conn_1$, which has an R-U function that evaluates to 3 for bandwidth 4 ($u_{1}(4) = 3$). Assume the total duration of the connection, $t_{1}^{max} = 10$ seconds of which 5 seconds have elapsed, denoted by $t_{i}^{age}$, and the allocated bandwidth during this time was $b_{1} = 4$. This means that the accumulated utility so far $u_{1}^{curr} = u_{1}(b_{1}) \times t_{1}^{age} = 3 \times 5 = 15$. At this time a new connection $conn_2$ is competing with the old one for the same bandwidth. Assume that for $conn_2$ the utility corresponding to bandwidth 4 is 5 ($u_{2}(4) = 5$). Because the $convex\_hull\_opt$ allocation algorithm is using the slopes (utility/bandwidth) of the R-U functions’ convex hulls to make decisions, and $3/4 < 5/4$, it chooses $conn_2$ in comparison to $conn_1$ and $u_{1}^{curr}$ is lost. Let’s see what is the utility gained by the system after the next 5 seconds: $u^{a} = u_{2}(4) \times 5 = 5 \times 5 = 25$. If the first connection had been kept, the utility would have been $u^{a} = u_{1}(b_{1}) \times t_{1}^{max} = 3 \times 10 = 30$, thus the swapping decision was wrong. Therefore, to replace an old connection with a new one, the utility generated by the new connection until the completion time of the old connection should be greater than the utility generated by the old connection during its entire life time (see shaded areas
4.5. DYNAMIC REALLOCATION

in Figure 4.2). In our example, \(conn_1\) should be swapped with \(conn_2\) only if \(u_2(4) \times 5 > u_1(4) \times 10\).

![Figure 4.2: Replacement opportunity](image)

In the above example we assumed that we have the choice only to swap \(conn_1\) with \(conn_2\), at the same bandwidth level. In general, we need to consider a connection that has multiple acceptable bandwidth levels (in its R-U function), of which one is the current allocation. To reflect the age, and thus the accumulated utility at the reallocation time point, we need to modify the R-U function of the existing connection as follows. Each allocation corresponds to a level \(l\) in the R-U function, where \(1 \leq l \leq k\), and \(k\) is the maximum number of levels. We denote the bandwidth at level \(l\) by \(b_i(l)\), and the corresponding utility by \(u_i(b_i(l))\). For instance, in Figure 4.3 (a): \(b_i(1) = 0, u_i(b_i(1)) = 0, b_i(2) = 2, u_i(b_i(2)) = 1\), etc. Let the already existing allocation level at a reallocation time point be \(j\). In Figure 4.3, \(j = 3\) and \(b_i(j) = 4\).

Here we explain how a modified R-U function for a class II connection is constructed and refer to Figure 4.3 (a) and (b) as an example. When constructing the modified R-U function, we can divide the allocation levels in two sets (corresponding to two situations). If the bandwidth allocation stays the same or is increased, \(u_i^a\) remains the same. Thus, for all these levels, the modified R-U function, denoted by \(u_i^{age}\), will be equal to the utility for the
Figure 4.3: Age modification for class I and II, with $t_i^{age} = 5$, $t_i^{max} = 10$, and actual bandwidth $b_i = 4$

existing allocation:

$$u_i^{age}(b_i(l)) = u_i(b_i(j)) \quad \forall j \leq l \leq k$$

Decreasing the bandwidth results in losing a portion (or all) of the connection’s accumulated utility so far. So we first compute the lost utility:

$$u_i^{lost}(l) = \left(u_i(b_i(j)) - u_i(b_i(l))\right) \times t_i^{age} \quad \forall 1 \leq l < j$$

Then we reduce the utility that can be accrued at the respective levels by modifying $u_i$ to $u_i^{age}$:

$$u_i^{age}(b_i(l)) = u_i(b_i(l)) - \frac{u_i^{lost}(l)}{t_i^{max} - t_i^{age}} \quad \forall 1 \leq l < j$$

Note that for $l < j$ there is a larger difference between two adjacent utility levels in $u_i^{age}$ compared to $u_i$; that is, the slopes of the segments of the convex hull of $u_i^{age}$ (for $l < j$) are steeper. That is, a reduction of the allocation can only be compensated by a higher utility gained from other connections. More precisely, the slope increase in the modified R-U function is exactly large enough so that, if bandwidth is reallocated to other connections, the newly
accepted (or improved) connections will generate not only a higher utility for the reallocated bandwidth, but in addition will also recover the utility lost by degrading this connection. The lost utility will be recovered during the interval $t_i^{\text{max}} - t_i^{\text{age}}$ (that is, before the time point at which the degraded connection would have released the bandwidth naturally).

For class I connections, any decrease in bandwidth means the connection is dropped (leads automatically to 0 bandwidth). The modified R-U function for a class I connection is presented in Figure 4.3 (c), if the original R-U function is the one depicted in Figure 4.3 (a). For this class, the zero-bandwidth level is calculated similarly to the class II case. Class III connections do not lose utility (waste resource) in case of a reallocation, thus their original R-U function needs no age modification.

Now the question becomes, do we assume that the real duration of every connection is known? Obviously this is too unrealistic to assume. In practice we have to resort to an estimate of a connection’s duration. The better the estimation of the connection duration, the more accurate the modification will be. This is because overestimating/underestimating the duration of a connection will underestimate/overestimate the importance of a bandwidth decrease for this connection. In Section 4.7.6 we further discuss how the system behaves in the absence of an exact knowledge of the duration.

### 4.5.2 Drop penalty influence

Class I and II connections are dropped (disconnected) whenever their momentary bandwidth becomes zero, since that connection yields no utility in the end. Class III connections should not be dropped because of bandwidth shortage, since they can recover at a later time, without penalty. Recall that each connection comes with its own drop penalty, $P_i^{\text{drop}}$. To reflect this sensitivity to disconnections, the R-U function is further modified (the effect is
additional to the age-dependent modification) as follows:

\[ u_i^{\text{drop}}(b_i(l)) = \begin{cases} 
  u_i^{\text{age}}(b_i(l)) - \frac{P_i^{\text{drop}}}{t_i^{\text{max}} - t_i^{\text{age}}} & \text{for } l = 1 \\
  u_i^{\text{age}}(b_i(l)) & \forall l \neq 1 
\end{cases} \]

Similar to \( u_i^{\text{lost}} \) in the previous subsection, in order to improve the accumulated utility, this penalty should be recovered before the natural end of the connection.

Note that the modification is only applied to the first level (where \( b_i(1) = 0 \)), because if bandwidth is not reduced to zero, the connection is not dropped. Figure 4.4 presents the further modification of the class II R-U function from Figure 4.3 (b) given a drop penalty \( P_i^{\text{drop}} = 8 \).

![Figure 4.4: Class II drop modification with \( P_i^{\text{drop}} = 8 \), \( t_i^{\text{age}} = 5 \), \( t_i^{\text{max}} = 10 \), and actual bandwidth \( b_i = 4 \)](image)

**4.5.3 Adaptation time influence**

Adaptation time, which reflects sensitivity to bandwidth fluctuations, is only applied to class III connections. Classes I and II are not subject to frequent reallocations (bandwidth increases are useless, decreases are bounded by the number of the R-U function levels).
As presented in Section 4.4.3, for a flexible class III connection, if reallocation is performed before the adaptation time passed since the last reallocation, \((I_i < A_i)\), the gained utility in this interval (normally \(u_i \times I_i\)) is diminished by \(P_{i,adap} = u_i \times I_i \times (A_i - I_i)/A_i\).

Each time there is a reallocation, the R-U function is modified to represent the sensitivity to the current reallocation frequency. If \(I_i < A_i\) and there is a change from the current allocation level, \(j\), then an adaptation penalty is incurred:

\[
u_i^{\text{adapt}}(b_i(l)) = \begin{cases} 
    u_i(b_i(l)) - \frac{P_{i,adap}}{I_i} & \forall l \neq j \text{ and } I_i < A_i \\
    u_i(b_i(l)) & \text{for } l = j \text{ or } I_i \geq A_i
\end{cases}
\]

An example of modifications depending on adaptation time is shown in Figure 4.5.

![Figure 4.5: Class III adaptation modification with \(A_i = 5\), \(I_i = 4\), \(P_{i,adap} = 2\), and actual bandwidth \(b_i = 4\)](image)

### 4.5.4 Algorithm overview

To summarise, Figure 4.6 presents a high-level version of our bandwidth (re)allocation algorithm. The algorithm is invoked periodically and independently for each cell of the network. Note that
all \( n \) connections presented in the cell (composed of ongoing connections, new connections, and handovers), are subject to reallocation. To include the effects of flexibility class and age, the drop penalty and adaptation time, in our allocation decision we construct the modified utility functions as described in the previous sections (the routine \( \text{age\_modify()} \) constructs \( u_{i}^{\text{age}} \), and so on). Finally, the modified utility functions are used as an input to the \( \text{convex\_hull\_opt} \) allocation algorithm presented in Section 3.5. Note that the utility function modifications are particular to the respective decision time point, and with each new invocation of the algorithm we apply them on the original utility functions. The new connections and handovers arriving in between two allocation rounds, are put in a queue that will be processed at the beginning of the next allocation period.

\[
\text{Bandwidth (re)allocation algorithm:}
\]

\[
\begin{align*}
\text{input:} & \quad \forall 1 \leq i \leq n: b_{i}, u_{i}, \text{class}_{i}, t_{i}^{\text{max}}, t_{i}^{\text{age}}, P_{i}^{\text{drop}}, A_{i}, I_{i} \\
\text{output:} & \quad \forall 1 \leq i \leq n: \text{new}\_b_{i}
\end{align*}
\]

//modify the R-U functions:

\[
\begin{array}{l}
\text{for } i := 1 \text{ to } n \text{ do} \\
\quad \text{if } \text{class}_{i} = I \text{ or } \text{class}_{i} = II \text{ then} \\
\quad \quad u'_{i} := \text{age\_modify}(u_{i}, \text{class}_{i}, b_{i}, t_{i}^{\text{max}}, t_{i}^{\text{age}}); \\
\quad \quad u'_{i} := \text{drop\_modify}(u'_{i}, \text{class}_{i}, P_{i}^{\text{drop}}, t_{i}^{\text{max}}, t_{i}^{\text{age}}); \\
\quad \text{if } \text{class}_{i} = III \text{ then} \\
\quad \quad u'_{i} := \text{adapt\_modify}(u_{i}, \text{class}_{i}, b_{i}, A_{i}, I_{i}); \\
\quad \quad \text{compute new allocation:} \\
\quad \quad (\text{new}\_b_{1}, ..., \text{new}\_b_{n}) := \text{convex\_hull\_opt}(u'_{1}, ..., u'_{n});
\end{array}
\]

Figure 4.6: The TARA (re)allocation algorithm

Some of the parameters used in the algorithm are presented in Table 4.1, and the following are added: \( \text{class}_{i} \) is the connection class, \( b_{i} \) is the current allocated bandwidth, \( \text{new}\_b_{i} \) the new allocation decision. As input the algorithm has all \( n \) connections that contend for bandwidth in this cell. For new connections the algorithm acts also as admission control, if \( \text{new}\_b_{i} = 0 \) the connection is rejected.
Utility accounting algorithm:

**Input:** \(1 \leq i \leq n: b_i, u_i, class_i, t_i^{age}, P_i^{drop}, A_i, I_i, b_i^{min}, period\)

**Output:** \(1 \leq i \leq n: u_i^{a}, u^{a}\)

**Algorithm:**

```
for i := 1 to n do
    if class_i = I or class_i = II then
        if (class_i = I and new_b_i \leq b_i) or (class_i = II and new_b_i = 0) then
            u_i^{a} := -P_i^{drop}; //rejected, apply drop penalty
        else //not rejected
            u_i^{a} := u_i(b_i^{min}) \times t_i^{age}; //set accum. utility so far
        end if
    end if
    if class_i = III then //update accum. utility so far
        u_i^{a} := u_i^{a} + u_i(new_b_i) \times period;
    end if
    if I_i < A_i then //apply adaptation penalty
        u_i^{a} := u_i^{a} - u_i(b_i) \times (A_i - I_i)/A_i;
    end if
    if new_b_i \neq b_i then //mark new reallocation
        I_i := 0;
    else I_i = I_i + period;
end if
//compute accumulated utility so far:
\(u^{a} := \sum_{i=1}^{n} u_i^{a}\).
```

Figure 4.7: The TARA utility accounting algorithm

To measure the utility accrued so far by the system, we use a utility accounting algorithm, presented in Figure 4.7. Note that this process is independent of the allocation algorithms, and is used for all the schemes compared in the evaluation section. Basically it periodically calculates the utilities according to the definitions presented in Section 4.4. New denotations used in this algorithm are \(b_i^{min}\), the lowest bandwidth granted in the connections’ lifetime, and \(period\), the running periodicity of the algorithm.

### 4.6 Evaluation setup

4.6 Evaluation setup

To evaluate the advantage of using utility-based characteristics of a connection we have compared our time-aware resource allocation scheme, TARA, with a recent adaptive allocation scheme that
addresses similar network problems, the RBBS. We begin with a short description of the RBBS proposed by El-Kadi et al. [2]. We then explain how we have reconstructed that algorithm in our simulation environment to make valid comparisons.

4.6.1 RBBS description

The Rate Based Borrowing Scheme successfully avoids some rejections by allowing bandwidth to be borrowed from already accepted connections. It uses the following strategy. Each connection that arrives in the system comes with a minimum ($min_i$) and a maximum ($max_i$) bandwidth requirement. The actual borrowable bandwidth ($abb_i$) is calculated as a fraction ($f$) of the difference between maximum and minimum bandwidth, $abb_i = f \times (max_i - min_i)$. $f$ is a cell-wide parameter to be applied to all connections. Another cell-wide parameter is $\lambda$ which is the number of equal shares the $abb_i$ is divided into. When there is not enough bandwidth available at a certain admission point, bandwidth is freed up by decreasing the allocation to all connections with one level (a share from the $abb_i$). Moreover, in order to provide a smooth change in bandwidth allocation, only one share from the borrowable part can be lent at the same time. RBBS divides connections in two classes. RBBS class I are considered real-time connections and a certain amount (e.g. 5%) of the cell bandwidth is reserved to be exclusively used by this RBBS class during handovers. This is because RBBS class I connections must have always at least $min_i$ bandwidth allocated all the time, otherwise the call should be dropped. RBBS class II applications are considered best-effort and can be handed over with any allocated bandwidth (greater than zero) in the new cell. Another differentiation is between the treatment of new connections and handed over ones. Dropping an ongoing connection is worse than blocking a new one. Therefore connections can be handed over at their minimum bandwidth requirement for RBBS class I (or greater than zero for RBBS class II). On the other hand, new connections (both RBBS class I and II) are only accepted if enough bandwidth is available to accommodate them at the same level as the whole
cell. When connections terminate or are handed over, the available bandwidth increases. If enough bandwidth becomes available, it is returned to the degraded connections, and the whole cell moves to a better QoS level.

The above work is very interesting because the authors take into consideration many of the characteristics of a modern network. They consider different traffic types, with different bandwidth requirements, multiple allocation levels, resource assurance classes, etc. On the other hand they do not use a quantitative performance metric (such as utility), that could glue such a complex system together and steer allocation, but use the usual performance metrics such as blocking/dropping probabilities, that are more suited for fixed allocation/single service systems (e.g. 2G). We will return to this issue later in the chapter.

4.6.2 Traffic and simulation parameters

For a good comparison with the RBBS, we use the traffic mix presented in Section 3.6. In addition, we have the following setup for the simulation environment.

Connections emerge from a MS following an exponentially distributed inter-arrival time with a mean of 15 minutes. All the 6 application groups in the traffic mix arrive with equal probability. Mobility is modelled in the following way: the time at which a MS changes cell (and requests a handover if a connection is ongoing) follows a geometric distribution starting from 60 sec and mean 300 sec, with equal probability to move in any of the neighbouring cells. Fluctuations of the wireless link, mentioned as a source of bandwidth variability in the introduction, have not been implemented in the simulator. Nevertheless, the random handover and new connection arrival, together with the different sizes and R-U functions of the connections ensure a very dynamic resource variability. We believe that since our system properly deals with this variability, the radio link variability can be dealt with analogously.

Our simulations were performed in a simulation environment described by Jonasson [57] and built on top of J-Sim, a component-based, simulation environment developed at Ohio State Univer-
We have simulated a hexagon cell-grid of 16 cells, $4 \times 4$, and a go-around world model to preserve uniformity in our grid. Each cell has a capacity of 30 Mbps.

For all the schemes the bandwidth allocation/reallocation has been performed with a period of 2 seconds. The drop penalty was set using the following formula $P^\text{drop}_i = 20\% \times u_i(b^\text{req}_i) \times t^\text{avg}_i$, where $b^\text{req}_i$ is the requested bandwidth, and $t^\text{avg}_i$ is the average connection duration (according to Table 3.1). Adaptation time was set to 5 seconds.

As our main performance metric we use the accumulated system utility ($u^a$) generated by the different connections in the system. The accumulated system utility is accounted independent of the allocation algorithm, and is calculated in the same way for all the simulated schemes, and according to Section 4.4.

### 4.7 Evaluation results

Figure 4.8 presents the accumulated utility generated by 5 allocation schemes (that will be described shortly) during one simulated hour. On the x-axis we have the arrival rate (number of new connections per second). The values in parenthesis represent the corresponding offered load as compared to the capacity of the cell. Thus $0.2(2.56)$ means that the offered load with an arrival rate of 0.2 was 2.56 times the maximum capacity of the cell. The offered load is calculated using the bandwidth requests of the connections.

For each of the arrival rates and for each bandwidth allocation scheme we conducted five different experiments (by changing the seed of the various distributions) and plotted the average value. The coefficient of variance ($CV$) was less than 0.06 in almost all of the cases ($CV = \sigma/\mu$, that is the standard deviation divided by the average). A similar statistical confidence applies also to the results presented in the forthcoming figures.
4.7. EVALUATION RESULTS

Figure 4.8: Accumulated utility

4.7.1 Comparison with basic maximisation

To see the impact of our class and age aware modifications, we have compared three flavours of TARA. TARA-normal and TARA-perf-est both use modified R-U functions as presented in Section 4.5. The difference is that for TARA-normal we have used the average connection duration (see Table 3.1) to estimate the duration of each connection when calculating the modifications (see Section 4.5), while for TARA-perf-est we used the real duration from the traffic generator. Thus, the latter provides the best possible case to hope for. Although the age-dependent modifications play an important role in our scheme, the difference between TARA-

- TARA-normal
- TARA-perf-est
- TARA-no-update
- RBBS-normal
- RBBS-friendly

Connection Arrival Rate

Total Utility per Cell
normal and TARA-perf-est in Figure 4.8 is marginal. It seems that in most of the cases, the difference between the real duration and the average value, is too small to result in the wrong decision (to decisively change the slopes of the modified R-U functions).

We have also simulated a version of TARA where the modifications of the original R-U functions are not performed, denoted as TARA-no-update. Basically, TARA-no-update is the convex hull opt allocation algorithm (see Section 3.5) invoked periodically. By not taking into consideration the connection classes, the dropping penalty and the adaptation time, TARA-no-update exhibits a 35% decreased system utility when working in areas where the offered load is between 1.3 and 2.6.

At high overloads (corresponding to 0.5 and 1 arrival rate) the applications with the lowest utility per bit, which belong to application group 3, class II, are all rejected at the beginning, and since the lowest utility per bit connections still accepted are now applications in group 6 class III, which can be put indefinitely on hold, TARA-no-update comes closer to the other two. This is an expected behaviour with a traffic in which the allocation borderline (the last bandwidth allocated) lies firmly within connection class III.

### 4.7.2 Comparison with RBBS

The results for RBBS have been plotted as RBBS-normal. There is a large difference between TARA and RBBS which amounts to 45% when the system gets overloaded with traffic. The main factor that contributes to this result is the absence of utility consideration by RBBS. While TARA is rejecting only low utility per bit connections, RBBS is rejecting a comparable amount from all application groups.

Besides the original RBBS we also used a slightly modified version of RBBS to make the comparison more favourable towards that scheme (shown as RBBS-friendly). The original RBBS may both lower and raise bandwidth for all connections. Hence, we modified RBBS not to replenish connections of TARA class II (because no utility is gained), and set the borrowable part of TARA
class I connections to zero. For both RBBS schemes, reserved bandwidth was $r = 5\%$, number of levels $\lambda = 10$, and borrowing factor $f = 0.5$ [2].

4.7.3 Effects of convex hull approximation

As a base allocation maximisation algorithm in the TARA scheme we have used the convex hull opt presented in Section 3.5. Approximating the utility functions with their convex hull frontier at allocation time can yield suboptimal results. However, in the proof of Lemma 3.5.1, we have shown that $U \leq U_{opt} \leq U_{aug}$, where $U = \sum_i u_i(b_i)$ is the system-wide utility generated by the $b_i$ allocation, and $U_{aug}$ is an upper bound calculated as $U_{aug} = \sum_i u'_i(b_i)$. By $u'_i$ we denote the convex hull frontier of the utility function $u_i$. Since we can easily compute $U$ and $U_{aug}$ at runtime, we can compute a better difference bound between $U$ and $U_{opt}$. Let’s denote it as $\Delta U_{opt} = U_{aug} - U \geq U_{opt} - U$. Note that this bound only pertains to the accuracy of convex hull opt.

During an experiment, the algorithm performs 900 allocation rounds during the simulated half hour. We computed the average $\Delta U_{opt}$ of these rounds. Since we test only convex hull opt, no age, adaptation time, or drop related modifications were applied to the original utility functions. Table 4.2 shows the results for different offered traffic loads (compared to the cell size).

<table>
<thead>
<tr>
<th>offered load</th>
<th>1.04</th>
<th>2.19</th>
<th>3.66</th>
<th>5.70</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg. #conns</td>
<td>17</td>
<td>29</td>
<td>42</td>
<td>61</td>
</tr>
<tr>
<td>avg. $U$</td>
<td>3164146</td>
<td>4355871</td>
<td>5140108</td>
<td>5925071</td>
</tr>
<tr>
<td>avg. $\Delta U_{opt}$</td>
<td>22681</td>
<td>41668</td>
<td>24637</td>
<td>7176</td>
</tr>
<tr>
<td>avg. $\Delta U_{opt} / avg. U$</td>
<td>0.0071</td>
<td>0.0095</td>
<td>0.0047</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

Note that all results are less than 1% away from the true optimum. This bound improves on the theoretical worst case bound given by Lemma 3.5.1. Moreover, this procedure can be used by service providers as an inexpensive tool for run-time profiling.
4.7.4 QoS per application group

So far we have presented the results only from the perspective of the total system utility. A more specialised view is presented in Table 4.3, the application groups on the x-axis refer to those in Table 3.1. We can observe that only connections that have the lowest utility efficiency are blocked (new connections) or dropped (ongoing connections). Since application group 6 is a class III connection, it can accept zero allocation situations, so there are no ongoing connections dropped in that case. Also, even at 13 times overload, most of the small, important application groups remain unscathed. Nevertheless it is important to note that the main goal of the system is to generate the highest utility and not to minimise the number of rejected/dropped connections.

Table 4.3: Statistics per application group at load 2.42 and 13

<table>
<thead>
<tr>
<th>application groups</th>
<th>load = 2.56</th>
<th>load = 13</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>accepted new</td>
<td>464</td>
<td>461</td>
</tr>
<tr>
<td>rejected new</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>rejected ongoing</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>allocation level (%)</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

4.7.5 Choice of performance metric

As the main performance metric, we use the accumulated system utility. Hence, we depart from the traditional call blocking probability (CBP) and call dropping probability (CDP) as performance metrics. We argue that they are obsolete in a system where the requested bandwidth of one connection might be only a small fraction of another connection’s demands, but both contribute equally in calculating CBP or CDP. The argument is confirmed by Figure 4.9, which shows the CBP of the simulations. The application group most blocked by TARA has a big bandwidth demand, and by blocking few of them a lot of bandwidth is saved for other connections. Since RBBS treats all connections equally it has to reject many more connections to equal the number of bits.
4.7. EVALUATION RESULTS

Although the aim of our algorithm is to maximise the utility and not to ensure a low dropping (or blocking) probability, dropping an accepted connection reveals a certain degree of miscalculation. Thus we present the CDP in Figure 4.10. Since TARA can also drop ongoing connections which are not handed over, we use a different formula for CDP.

\[
CDP = \frac{\text{rejectedOngoing} + \text{rejectedHandovers}}{\text{acceptedNew} + \text{acceptedHandovers}}
\]

Even without reserving a certain amount of bandwidth to be used exclusively for handovers (RBBS reserves 5% for this purpose), TARA-normal and TARA-perf-est are able to keep the number of drops quite low. Handovers are not regarded as new connections in the cell where they are handed over. We want to emphasise here that the increased importance due to the aging mechanism provides a “natural” differentiation between handovers and new connections, and the algorithm does not have to use some kind of forced differentiation mechanism in their treatment.
The experiments show that the aging mechanism, the dropping penalty, and the flexibility of class III connections are able to protect handovers as well as other ongoing connections, from being dropped. The consequence of not taking into consideration these factors is shown in the plot of TARA-no-update. While blocking less connections, it is dropping more than TARA-normal. The effects on the accumulated utility were already presented in Figure 4.8.

4.7.6 Connection duration estimation

Although the age-dependent modifications play an important role in our scheme, the difference between TARA-normal and TARA-perf-est in Figure 4.8 is marginal. It seems that in most of the cases, the difference between the real duration and the average value, is too small to result in wrong allocation decisions.
Thus we ran some experiments with a traffic mix more sensitive to duration estimation errors (i.e. containing no flexible, class III applications). The results are plotted in Figure 4.11. Using the average duration of the distribution (TARA-normal) still gives results close to the perfect estimation. They are both contrasted by the TARA-random curve. In the “random” case the estimation value for the duration is chosen randomly between 0 and 10 hours. The results show that if the duration of the connection is approximated by the average duration, the differences in the system utility are quite small compared to a perfect estimation.

![Figure 4.11: Differences due to estimated duration](image)

### 4.7.7 Complexity considerations

From a computational complexity point of view, the convex hull optimisation algorithm that we use, has a complexity of $O(nL \log n)$, where $n$ is the number of ongoing and new connections, and $L$ is
the maximum number of utility levels of an R-U function. The utility function modifications that we introduce have the complexity of at most $O(nL)$, since they have to manipulate each level in the R-U function. The RBBS algorithm has a worst case complexity of $O(n)$, since it has to access each connection when borrowing bandwidth. When borrowing does not occur, that is until free bandwidth is depleted, the algorithm just serves the new incoming connections ($O(1)$). The above analysis shows that considering only algorithmic computations, the RBBS has a favourable complexity compared to ours. However, the bandwidth reallocations are the ones that might impose a heavy burden on the system due to executions of control functions and the associated signalling. Since we expect that the reallocation overhead is more important than the computational complexity, we specifically study the trade-off between utility optimisation and reallocation overhead.

### 4.8 Infrastructure overhead

By analysing the applications’ requirements and modifying the R-U functions, the allocation algorithm takes account of the effect of reallocations from an application point of view. However, the scheme ignores the overheads created by reallocations on the system infrastructure. First, there is the increase in signalling. Every time the allocated bandwidth changes, the radio network controller (RNC), where the radio resource management decisions are taken, has to send the bandwidth reallocation decision to both source and destination of the respective connection. Second, processing capacity is demanded by control functions used in conjunction with a bandwidth change (e.g. channel switching: both type and rate, setup/release connection, power control). A more detailed relationship between control functions and CPU load in a RNC, together with a lower-level CPU overload protection mechanism are presented in Chapter 7.

To measure the infrastructure overhead we count the number of changes occurring in the respective cell at each reallocation point. Note that the overhead is proportional to the number of bandwidth
changes and does not depend on the amount of bandwidth change. Figure 4.12 compares the number of bandwidth reallocations for TARA and RBBS for different connection arrival rates (i.e. offered load).

As in the previous graphs, the x-axis depicts the connection arrival rate (CAR) and the values in parenthesis represent the offered load compared to the capacity of the cell. While the system is not overloaded (before 0.1 on the x-axis) the number of changes is low. It begins to grow (approximately linearly) when the system is overloaded. Under initial overload (2.56 cell capacity) the difference between TARA and RBBS is around 50%. The proportion slowly decreases as the system gets heavily overloaded with traffic. It is explained by the fact that, RBBS is borrowing bandwidth from all connections in the system (and thus reallocating for all), while TARA is degrading only the lowest performers.
Our next concern is how to modify the allocation maximisation algorithm so that we decrease the number of bandwidth changes while still keeping a high total system-wide utility. Or the related question, what is the relationship between the number of reallocations and the performance (generated utility) of the system?

### 4.8.1 Reallocation control

The magnitude of overhead generated by bandwidth reallocations and their effects on the system-wide utility greatly depends on the system implementation and also on the run-time environment. Thus, it is difficult to analytically associate a penalty tag to a reallocation to use it directly in the utility maximisation algorithm. Nevertheless, it is clear that by reducing the number of reallocations the overhead is proportionally reduced. To this end, we propose and evaluate an enhanced bandwidth allocation algorithm that minimises the number of bandwidth changes while keeping the utility at a predetermined level. The algorithm, presented in Figure 4.13, replaces the original \textit{convex hull opt} algorithm in TARA (see the (re)allocation algorithm in Section 4.5.4).

The core idea is as follows: each period we first compute a bandwidth allocation that maximises system utility regardless of the number of generated bandwidth changes (using the original \textit{convex hull opt}). This generates a list of changes to be performed. Then we discard changes from the list as long as the utility remains above a minimal acceptable value. The minimal acceptable utility is proportional to the maximal utility allocation and is calculated using a control threshold parameter ($thresh$). That is, if $thresh = 95\%$, the utility generated in each period is never lower than 95\% of the maximum attainable. The pseudocode in Figure 4.13 describes the algorithm.

Figures 4.14 and 4.15 show the performance of the algorithm (measured in the number of reallocations), when used with different threshold settings. All the other parameters used in the experiments are identical to the ones presented in Section 4.6. Figure 4.14 presents the dependency between reallocation count and offered traffic load. As the traffic load increases the differ-
MinChangeAlloc Algorithm

input: \{c_1, c_2, ..., c_n\} //set of connections
thresh //system utility threshold

output: \{b'_1, b'_2, ..., b'_n\} //newly allocated bandwidth

begin
initialise: u_i := R-Ufunction(c_i) //standard R-U function
b_i := bandwidth(c_i) //current bandwidth

\{b'_1, b'_2, ..., b'_n\} := convex_opt(u_1, u_2, ..., u_n) //maximum utility

G := \{ c_i | b'_i > b_i \} //connections gaining bw.
L := \{ c_i | b'_i < b_i \} //connections losing bw.

eff_i := \left| \frac{u_i(b'_i) - u_i(b_i)}{b'_i - b_i} \right| //the efficiency criterion

G_a := sort_a(G) //sort efficiency-ascending
L_d := sort_d(L) //sort efficiency-descending

fbw := B_{\text{max}} - \sum_{i=1}^n b'_i //calculate initial free bw.

while G_a \neq \emptyset \land L_d \neq \emptyset loop
identify a bw. decrease and a set of covering increases:
identify c_l = head(L_d)

if \exists a minimum prefix PG_a of G_a such that
b_l - b'_l \leq \sum_{c_j \in PG_a} (b'_j - b_j) + fbw then
//resulting utility if we give up these reallocations:
u' := u' - \sum_{c_j \in PG_a} (u_j(b'_j) - u_j(b_j)) - u_l(b'_l) + u_l(b_l)
if u' < thresh \times u_{\text{max}} then exit loop
//else really give these reallocations up:
fbw := fbw - \sum_{c_j \in PG_a} (b'_j - b_j) - b'_l + b_l
b'_l := b_l; \forall c_j \in PG_a b'_j := b_j
G_a = G_a - PG_a; \ L_d = L_d - \{c_l\}
else exit loop
end loop

return \{b'_1, b'_2, ..., b'_n\}

end

Figure 4.13: Reallocation control algorithm
ence between the allocation algorithms increases too. For instance, when \( \text{CAR} = 0.1 \) (the offered traffic load is 130%), the number of reallocations with \( \text{thresh} = 80\% \) is 25\% lower than for TARA-normal. With the new algorithm when \( \text{CAR} = 0.5 \) (the offered load is 650\%) the number of reallocations drops with 68\%. Note that for a given threshold value (representing the minimal acceptable utility) the actual utility gained by the system can be much higher. A better perspective on the dependency between reallocations and the generated utility is presented in Figure 4.15. While for lower loads, the increase in reallocations is not so big, for higher traffic loads (higher connection arrival rates), the number of reallocations strongly increases as we get closer to maximum utility. Thus we can greatly diminish the number of bandwidth reallocations by slightly lowering the utility expectations.

Figure 4.14: Performance of reallocation minimisation algorithm. View (a)
4.9 Summary

In an open, dynamic system there is a trade-off between optimisation and provisioning. A resource allocation decision might promise maximal utility at a certain time point, but as new requests arrive and old requests depart, a reallocation could improve the utility of the system. The question arises: should resources be reallocated or not? A reallocation might accept connections with higher utility, but might also break ongoing QoS contracts. The novelty of our approach is that we combine these choices in a consistent manner. We synthesise the consequences of potential reallocations for different classes of applications, and use this information in our periodic allocation/reallocation strategy.

In this chapter we have presented an admission control and resource allocation scheme to be used in a future generation mo-
mobile network. The scheme is based on an allocation algorithm that aims to maximise system-wide utility, having the utility of each connection specified by a bandwidth dependent utility function. To suit the dynamic nature of the environment, where constant reallocations are required, we identified the effects of reallocations on different connections. Based on their sensitivity to reallocations, connections have been divided into three classes: non-flexible, semi-flexible, and fully flexible. Connections are also differently affected by disconnections and have different sensitivity to allocation fluctuations. An implicit but very important factor is the age of the connection, since it represents the time during which resources have already been invested.

While the application here might seem specific, we believe that a similar approach can be adapted for other open, dynamic environments (e.g. the link capacity of an Internet provider) or other resource types.

To validate our approach, the algorithm has been tested against a baseline that does not take account of the above factors. We have also compared it with a recent adaptive allocation scheme (RBBS), that does not use a value-based approach. Our approach shows significantly increased performance as expressed by the system-wide accrued utility. Another advantage is that the treatment of handovers is consistent with that of other ongoing connections, by taking into account their age-related increased importance when allocating bandwidth in the new cell.

After identifying a (re)allocation scheme based solely on application preferences, we considered the overhead that the scheme generates in terms of demands on other resources in the system. For example CPU time utilisation and signalling traffic increases when executing the reallocation decisions. We have chosen the number of bandwidth reallocation as a metric for characterising such demands and shown that we can greatly reduce the strain on the system with only a small decrease in the generated utility. Consequently, this new algorithm could be employed to avoid overloading the infrastructure.
Chapter 5

Modelling non-critical resources as cost

Multi-resource allocation optimisation is way more complex than single resource allocation, and rough heuristics must be employed for time-efficient solutions [21]. Nevertheless in many practical situations, only one of the resources is a major limiting factor in the system. Therefore, we propose an approach that centres around an accurate model for the bottleneck resource allocation, but also takes a rough account of the other resources in the system.

We use the resource-utility functions presented in Chapter 3 to characterise the dependency of utility on the amount of allocated bottleneck resource. While still important, other resources are considered non-critical in comparison with the bottleneck resource. Thus, they will appear as unconstrained in the allocation problem. Nevertheless there is an implied cost in using these resources and we model them using a cost function that is subtracted from the utility gained by the connection. Finally, by combining the utility function of the bottleneck resource and the cost function we can at the same time optimise the usage of the bottleneck resource while minimising the usage of non-critical resources. If applicable, this resource allocation model can use the computationally simpler algorithms used in single resource allocation (as
opposed to multi-resource models), while also improving the usage of the “non-critical” resources.

As an example of such a system, in this chapter we present the case of bandwidth allocation in a hybrid (ad hoc over cellular) network setting. Hybrid wireless networks propose to combine the advantages of both an ad hoc and a cellular network, for instance extending the reach of the cellular network by ad hoc paths while using the stability and hierarchical structure of the cellular network. In this setting a mobile station can choose to connect to the core network via different base stations. Incentives for relaying users are modelled as cost and included in the allocation problem. A distributed algorithm that has far lower computational overhead and accrues at worse 12% less than the utility of an optimal centralised solution is proposed. Comparative studies are made to show effective load balancing and crash tolerance in the presence of a high traffic overload.

5.1 Overview

In Future Generation wireless networks, diverse wireless technologies such as Cellular, WLAN, and Bluetooth will proliferate in different edges of the Internet and complement each other to provide untethered multimedia services and seamless visits to the IP-core network. One of the challenges in the hybrid wireless networks, is optimal resource management of diverse radio resources, from the perspectives of both the users and the service provider. Hybrid network radio resources often include “cellular” capacity (licensed frequency spectrum centred around fixed base stations) in addition to “ad hoc” capacity (unlicensed frequency spectrum limited by interference local to each mobile). Furthermore, hybrid network models, that employ user equipment to serve as mobile relays, must include resulting usage costs into resource management.

Most wireless access technologies are deployed in either infrastructure based cellular mode or infrastructure-less ad hoc mode. While each access mode was initially designed with distinct characteristics, many recent efforts are underway to define hybrid net-
5.2. RELATED WORK

works, that combine the advantages of both access modes [62, 63, 64, 65]. These approaches to hybrid networks can be classified as either “ad hoc over cellular” [65, 66, 67, 68] or “cellular over ad hoc” [69, 70]. The “ad hoc over cellular” approach aims to “stretch” the reach of cellular networks (reducing infrastructure costs), extend mobile battery life by reducing link length, and integrate high speed access, global coverage and roaming support into a single seamless system. These concepts motivated the “opportunity driven multiple access” (ODMA) option in 3GPP [62] as well as next generation A-GSM [64].

In this chapter, we extend the optimal resource allocation and admission control scheme in TARA for an ”ad hoc over cellular” context. The bottleneck resource is the bandwidth of the cellular link, and consumption of energy and processing power are modelled as a cost function. Two resource allocation algorithms result from this investigation. First, a centralised optimal allocation algorithm based on linear programming is formulated. Second, a distributed heuristic algorithm is formulated that attempts to perform close to the optimal solution with considerably lower runtime complexity. The simulation analysis, using an extended cellular TARA simulator, illustrates the performance gains in ”ad hoc over cellular” hybrid networks. It demonstrates the capability of the proposed heuristic algorithm to efficiently utilise resources in the hybrid radio context and provide benefits such as load-balancing and fault tolerance.

5.2 Related work

Several “ad hoc over cellular” approaches such as MCN [65] and iCAR [66] use relays to overcome cellular shortcomings, such as limited spatial coverage, low bit rates, and a high bit cost for data services. Relays, being either static infrastructure or other mobile stations (MS), form a virtual overlay for congestion mitigation and alternate routing to extend and improve coverage of the cellular base stations (BS) [64, 71, 72]. In these hybrid networks, admission
control (AC) and bandwidth allocation (BA) schemes for resource management are necessary to ensure QoS guarantees.

Work on resource management for cellular networks often focuses on management of licensed frequency spectrum local to each base station. For example, the authors in [73] present a novel adaptive bandwidth allocation scheme with dynamic local estimation of changing traffic parameters, and a probabilistic control policy for high channel utilisation. The work in [2, 49] employs bandwidth borrowing and degradation as part of AC with each connection request submitting acceptable max and min resource requirement. In [74] an AC algorithm is proposed that uses controlled QoS degradation of on-going calls to manages a tradeoff between resource allocation of on-going calls and new calls. A similar tradeoff is managed by an AC algorithm proposed in [75] using Guard Channel policies. AC schemes proposed in [76] consider both “non-prioritised” schemes in which the BS made no distinction between new and HO calls, and two “priority oriented” schemes that allow queuing of handover calls.

Managing the local interferences, generated by consumption of unlicensed frequencies, while maintaining end-to-end multihop connections is a general concern in ad hoc networks. For example, in [77] a contention-aware AC is proposed which attempts to support QoS guarantees by limiting the number of connections allowed within a neighbourhood of nodes. A distributed AC algorithm is introduced in [78] that is based on the concept of a “service curve” to reflect the status of the network (number of active nodes, activity index and contention status). An ad hoc node wanting to establish a new connection must compare the “service curve” with a predefined universal performance threshold curve for QoS purpose.

An emerging focus on resource management for hybrid networks indicates that new mechanisms are needed to manage the heterogeneous air interfaces inherent in hybrid networks [65][66]. Admission control and bandwidth allocation schemes are needed that simultaneously consider the radio capacity centred around base station access points and radio capacity centred around the
5.3 ALLOCATION IN HYBRID NETWORKS

MS, and give further consideration to the impact of using MS relays as part of hybrid network infrastructure.

Resource management competition strategies between network providers are introduced in [79]. A joint admission control and rate control framework for CDMA networks is proposed. User and system utility functions are separately defined, and a non-cooperative game is formulated where providers compete for customers.

5.3 Allocation in hybrid networks

In this section we explain the system model used for our utility maximisation scheme. We start from a classic cellular network model, where in each cell a base station (BS) services the mobile station (MS) inside the covered area. MSs connects to the BS using the direct cellular wireless link. In addition, we assume each MS is equipped with a second wireless interface that can be used to connect to other MSs in an ad hoc manner. We consider the two spectra (cellular vs. ad hoc) to be in different bands, the cellular using a narrower, highly regulated band while the ad hoc belongs to a broader, reusable, unregulated band. Thus, there is no interconnection/interference between the two bands.

5.3.1 System model

At a certain point in time, a MS can connect to its host cell directly through the cellular link, or it connects first via several different ad hoc paths to other MSs, and further through the cellular interface to the host base stations of the last MS in the path. Figure 5.1 presents an example. The ad hoc network serves only as an extension for the cellular network, with most of the functionality (allocation, security, billing, etc.) located in the nodes of the cellular network.

Regarding the bandwidth allocation problem, we make the following observation. The bandwidth of the ad hoc network is usually more than one order of magnitude greater than the bandwidth of the cellular network. For example, today the bandwidth of a
3G base station is 2 Mb/s (with 10Mb for HSDPA mode) while the bandwidth of 802.11g is 40Mb/s (with 802.11n > 100 Mb/s). Thus, the bottleneck resource of the system is the cellular link bandwidth, which makes the ad hoc links bandwidth virtually unrestricted. We can regard the amount of traffic on a certain path to be restricted only by the bandwidth allocated on the cellular link. To describe the dependency of the utility function on the bandwidth allocation over the cellular link we use the full utility model of TARA, presented in Chapters 3 and 4 (i.e. utility functions, flexibility classes, drop penalties and adaptation times).

Even though we consider the bandwidth of the BS as bottleneck, in an optimised allocation we have to consider the effects of using the ad hoc paths. First, there is increased resource consumption on the relaying MS such as battery energy and processing power. Thus, we assume that users would appreciate some incentive for letting other connections use their MS. Second, there is the problem of the weaker QoS offered by the ad hoc route. Delay increases with hop count. Moreover, an ad hoc path might get disconnected due to mobility. We model these relay costs and QoS losses with the help of a path dependent cost that is proportional to the number of ad hoc hops and the amount of traffic sent on that path. From a pricing viewpoint, we can regard the utility functions as proportional to the rates the user is willing to pay for a certain connection. In the same manner, incentives proportional to the per hop costs could be regarded as reimbursements to the owner of the MS used as relay.

In order to establish an end-to-end connection the algorithm must choose first among a set of reachable BSs, where for each BS there might be several ad hoc paths available. Taking into account the above cost model, it is obvious that the shortest (in the number of hops) ad hoc route to a BS has also the lowest costs. Thus, the path choice in our scheme consists of two phases. First, a shortest path first (SPF) routing algorithm (such as AODV [80] or DSR [81]) is employed to find the best paths from an MS to a set of near BSs. A BS is considered “near” if a shortest path exists, given that the hop-count does not exceed a certain threshold value.
5.3. ALLOCATION IN HYBRID NETWORKS

Second, the allocation algorithm will use this set of paths in the optimised allocation.

Thus the model of Figure 5.1 can be condensed to the one presented in Figure 5.2, where the links represent potential paths from the MSs to the BSs. The numbers on the links in Figure 5.2 represent the number of relays, to which the cost/bit of that path is proportional. A direct cellular connection has zero relays.

5.3.2 Optimal bandwidth allocation

Assuming each connection has an attached utility function $u_i$, let’s calculate the system-wide utility we obtain with a certain
bandwidth allocation (at a certain time point). When we refer to a connection we imply an end-to-end, OSI transport layer connection. In the optimal allocation, the packets for a connection (e.g. between an application on a MS and its server in the core network), could be sent over several paths through several BSs. Let there be $n$ active MSs that connect to $m$ BSs. Assume $X_{ij}$ the amount of bandwidth allocated to a connection from MS $i$ over the $ij$ path (if such a path exists) to BS $j$. The cost the system incurs over a path $ij$ is modelled as $C_{ij} = c \times h_{ij} \times X_{ij}$, where $c$ is a cost constant that represents the cost/bit/hop and $h_{ij}$ is the number of hops. If a direct connection over a cellular link is possible, then $h_{ij} = 0$. The existence of a path $ij$ and its hop-count, $h_{ij}$, is given by the underlying SPF routing algorithm that finds the shortest paths between MS $i$ and the set of near BSs $j$.

The system-wide utility for a certain allocation is the sum of the utility generated by all connections minus the sum of the costs over all paths.

$$U = \sum_{i=1}^{n} u_i \left( \sum_{j=1}^{m} X_{ij} \right) - \sum_{i,j=1,1}^{n,m} c \times h_{ij} \times X_{ij} \quad (5.1)$$

Therefore, to derive the optimal value for bandwidth (now represented by a variable $x_{ij}$) we have to solve the following maximisation problem:

$$\text{Maximise } U = \sum_{i=1}^{n} \left( u_i \left( \sum_{j=1}^{m} x_{ij} \right) - \sum_{j=1}^{m} c \times h_{ij} \times x_{ij} \right) \quad (5.2)$$

subject to:

$$\sum_{i}^{n} x_{ij} \leq X_{j}^{\text{max}} \quad (5.3)$$

$$x_{ij} \geq 0 \quad (5.4)$$

where $X_{j}^{\text{max}}$ is the maximum bandwidth available in the cell $j$. If there are no paths between a MS and a BS then the corresponding term is excluded from all the $j$-indexed sums.
5.3.3 Linear programming formulation

Lee et al. [20] show that maximising $\sum_i u_i(x_i)$ subject to $\sum_i x_i < X^{max}$, where $u_i$ is a discrete R-U function and $X^{max}$ is the maximum available resource, is an NP-hard problem. They propose a near-optimal approximation algorithm that works with the convex hull of the R-U functions. If we denote $x_i$ to be the allocated bandwidth to connection $i$ over all possible paths, $x_i = \sum_{j=1}^{m} x_{ij}$, and we set $j = 1$ and $c = 0$ in equation (5.2) we observe that their problem is an instance of our problem, which makes our maximisation problem also NP-hard. To make the problem tractable, we approximate $u_i$ with its convex-hull frontier $u'_i$.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{utility_function.png}
\caption{Segment of a utility function}
\end{figure}

In Figure 5.3 we present a utility function (solid line), its convex hull (dashed line) and highlight the $k = 2$ segment of the convex hull. While $u'_i$ is not linear, it consists of several segments $(1, ..., s)$ that are piecewise linear, e.g. three in Figure 5.3. For a segment $k$ of $u'_i$ let $x_{ik}$ be the allocated bandwidth. The efficiency of the segment being $e_{ik} = \frac{U_{ik} - U_{ik-1}}{B_{ik} - B_{ik-1}}$, its corresponding incremental utility would be $e_{ik} \times x_{ik}$. Then, for a certain allocation, we have

$$u'_i(x_i) = \sum_{k=1}^{s} e_{ik} \times x_{ik} \quad (5.5)$$

where $x_i = \sum_{k=1}^{s} x_{ik}$. To transform our maximisation problem into a linear programming form, we would like to replace $u'_i(x_i)$ with $\sum_{k=1}^{s} e_{ik} \times x_{ik}$. However, if we regard the left and right sides of equation (5.5) as functions, the function on the right hand side is less constrained. Therefore, we need to add the following two
constraints. First, the bandwidth of a segment is restricted by its maximum as specified in the utility function, that is \( x_{ik} \leq X_{ik}^{\text{max}} \), where \( X_{ik}^{\text{max}} = B_{ik} - B_{ik-1} \). Second, a higher level (segment) cannot be used if the earlier levels are not fully used. That is, if \( x_{ik} > 0 \) then for all \( l < k \), \( x_{il} = X_{il}^{\text{max}} \).

Having expressed \( x_i \) with utility-segment related terms \( x_{ik} \), we can go back to the multi-path formulation in equation (5.2). The bandwidth allocated to \( x_{ik} \) can be distributed over a number of paths. Thus, \( x_{ik} = \sum_{j=1}^{m} x_{ikj} \) and \( x_{ikj} \) is the bandwidth allocated to the utility segment \( k \) of connection \( i \) on path \( ij \).

Now, the maximisation problem based on \( u'_i \) can be formulated as:

\[
\text{Maximise } U' = \sum_{i=1}^{n} \left( u'_i \left( \sum_{j=1}^{m} x_{ij} \right) - \sum_{j=1}^{m} c \times h_{ij} \times x_{ij} \right) = \\
\sum_{i=1}^{n} \left( \sum_{k=1}^{s} (e_{ik} \times \sum_{j=1}^{m} x_{ikj}) - \sum_{k,j=1,1}^{s,m} c \times h_{ij} \times x_{ikj} \right) = \\
\sum_{i,k,j=1,1,1}^{n,s,m} (e_{ik} - c \times h_{ij}) \times x_{ikj} \tag{5.6}
\]

subject to:

\[
\sum_{i,k=1,1}^{n,s} x_{ikj} \leq X_{j}^{\text{max}} \tag{5.7}
\]

\( x_{ikj} \geq 0 \tag{5.8} \)

\[
\sum_{j=1}^{m} x_{ikj} \leq X_{ik}^{\text{max}} \tag{5.9}
\]

\( \forall i, k, l \) if \( \sum_{j=1}^{m} x_{ikj} > 0 \) and \( l < k \) then \( \sum_{j=1}^{m} x_{ilj} = X_{il}^{\text{max}} \tag{5.10} \)

\textbf{Lemma 5.3.1} Any solution to equation (5.6) subject to conditions (5.7) to (5.9) respects condition (5.10).

\textbf{Proof} Let’s assume the opposite, which means that in such a solution there are two utility-segments \( k < l \) of a connection \( i \)
where $\sum_{j=1}^{m} x_{ikj} < X_{ik}^{max}$ and $\sum_{j=1}^{m} x_{ilj} > 0$. We know that $x_{ik} = \sum_{j=1}^{m} x_{ikj}$. Let $y = \min (X_{ik}^{max} - x_{ik}, x_{il})$. The utility generated by segments $k$ and $l$ is $U_{kl} = e_{ik} \times x_{ik} + e_{il} \times x_{il}$. We then take $y$ from segment $l$ and allocate it to segment $k$. Then $U'_{kl} = e_{ik} \times (x_{ik} + y) - e_{il} \times (x_{il} - y) > U_{kl}$ because $e_{ik} > e_{il}$ for a concave function such as the convex hull. The allocations for other segments being equal, this means that the original allocation was not maximal, so we arrive at a contradiction.

Thus, linear programming optimally solves the bandwidth allocation problem of equations (5.6) to (5.9).

As a baseline in our comparisons we thus adopt the LP approach. However, we are aware of the drawbacks of the approach that justify looking for a better solution. These are:

- **Centralised allocation.** The LP algorithm needs to know the state of all the MSs and BS and paths in the whole network to reach an optimal allocation. This is unrealistic for a large network.

- **Time complexity.** As will be presented in Section 5.5.3 the LP algorithm is prohibitively computationally intensive for an online allocation.

- **Signalling/control overhead.** The LP solution might spread the allocation of a connection over several paths through different BSs, which can increase both the logistical overhead and the contention on the MAC-layer.

Note that the topology of the network changes in time, ongoing connections end, and new ones are created. Old resource allocations can break or become suboptimal. To address this, we run our (re)allocation algorithms periodically. This bounds the reallocation rate in the system, even if the rate of events (traffic and topology changes) is much higher. The only disadvantage is that new and rerouted connections must wait until the next allocation time, to receive new resources.
5.3.4 Hybrid-heuristic algorithm

To solve the hybrid resource allocation problem in a distributed manner with less complexity (and overhead) we have devised the following heuristic algorithm. The algorithm can be divided in two parts.

- The core allocation algorithm is used to independently allocate resources for each BS. It compares all the different connections requesting resources at the given BS, and the most utility-efficient connections are chosen, taking into account also the incurred relaying costs. That is, at BS $j$, for each utility-segment $k$ of a connection $i$ a new core-efficiency, $e_{ikj}$ is computed by subtracting the relaying costs of path $ij$ from the original efficiency $e_{ik}$. Thus, $e_{ikj} = e_{ik} - c \times h_{ij}$. Then bandwidth is allocated in decreasing order of $e_{ikj}$. Let $comp_j$ be the lowest core-efficiency of an accepted (i.e. a non-zero allocation) utility-segment of a connection, $comp_j = \min(e_{ikj} \mid x_{ikj} > 0)$. This parameter characterises the level of the competitiveness of the connections requiring bandwidth at BS $j$, and will be used by a MS to choose the least competitive BS from its point of view. Note that $comp_j$ depends on both BS capacity and the importance of the contending connections. Similar to the LP algorithm, the “core allocation” part is invoked periodically, to keep the allocation updated.

- The path choice algorithm compares the paths to different BSs returned by the underlying routing algorithm, and chooses only one to carry the entire connection. The path choice algorithm will choose the connection to the BS where it assumes it has the highest chance to be accepted. To achieve this, it asks all near BSs about their $comp_j$ parameter. Intuitively, the BS with the lowest competitiveness during last allocation round should give the highest chance of accepting the connection. Nevertheless, the efficiency of the connection will be diminished by the path cost to the respective BS. Therefore among all the paths to possible BSs,
the algorithm chooses path $ij$ with $\min_{j=1}^{P} (comp_j + c \times h_{ij})$, where $p$ represents the number of BSs near to MS $i$. The “path choice” algorithm is event-triggered: a) at the arrival of a new connection, b) when the current path cannot be sustained anymore (due to mobility, fading, failures). In this new context of ad hoc paths, handovers can be of two types: the traditional, when the MS uses the direct cellular link and moves out of the BS reach area, or, when one of the relays in the ad hoc path moves out of range. In either case, handovers are treated similar to ongoing connections, i.e. the already accrued utility is taken into account.

The path cost related component in the “path choice” algorithm also prevents oscillations in the system. By oscillations we mean that at a certain point in time a cell is the target of most of the new connections/handovers, while at the next point all the load is directed to another cell. Due to the cost differences of different paths however, the MSs tend to connect to closer BS if the competitiveness factors are roughly the same.

5.4 Evaluation setup

To evaluate the behaviour of our hybrid network resource allocation scheme we use the same traffic mix used for testing the cellular TARA (Chapter 4). The traffic types and their associated utility functions are presented in Section 3.6.

Our simulations were performed in an extension of the simulation environment used for TARA, by allowing a MS to connect to a neighbouring BS, if an ad hoc path is available. For the linear programming part, we have used the java package from the operation research objects collection (OR-Objects) [82].

Connections arrive on the MSs following an exponentially distributed inter-arrival time with a mean of 15 minutes. All the 6 application groups arrive with equal probability. Mobility is modelled in the following way: the time at which a user moves in a new geographical cell follows a geometric distribution starting from 60
sec and mean 300 sec, with equal probability to move in any of the neighbouring cells. If the MS uses an ad hoc path, a handover is triggered when the current path gets disconnected. To simulate this, we employ a simple but efficient ad hoc path generation mechanism. Following a geometric distribution with the mean of 300 sec it triggers in a MS a path renewal that discards the old paths and creates \( p \) new paths to randomly chosen neighbouring BSs. To each of the paths, a random hop count between 1 and \( \max_{\text{hop}} \) is attached. The experiments have been conducted with \( p = 3 \) and \( \max_{\text{hop}} = 4 \). The cost/bit/hop of using an ad hoc path has been set to \( c = 0.02 \) (unless otherwise stated).

We have simulated go-around world model to preserve uniformity in our grid. Each cell has a capacity of 30 Mbp/s. For all experiments the bandwidth allocation/reallocation has been performed with a period of 2 seconds. Further QoS controlling parameters have been kept unchanged from our previous TARA experiments.

5.5 Experimental results

In this section we test the performance of the hybrid-heuristic algorithm, to show how close it comes to the optimal LP algorithm. Furthermore, we use also the pure-cellular TARA allocation. The comparisons are performed using three key scenarios to expose the characteristics of the hybrid network. The first scenario simulates uneven loaded cells (hot-spots) and we test the load balancing features of the hybrid network. The second scenario exposes the fault tolerant capabilities as we simulate a BS failure, while in the last scenario we test a uniformly balanced setting, where the hybrid network should not perform better than a pure cellular setting. We then go on to show QoS differentiation properties and algorithm timing overheads.
5.5. EXPERIMENTAL RESULTS

5.5.1 Accumulated utility as performance

As our main performance metric we use the time-accumulated system utility, generated by all the connections in the system. We show the behaviour of the system when subjected to increased traffic loads, as marked on the x-axis. The numbers represent the offered traffic load compared to system capacity (e.g. 2.42 means that the offered load traffic is on average 2.42 times the systems capacity). For each of the offered loads we conducted five different experiments (by changing the seed of the various distributions) and plotted the average. The coefficient of variance was less than 0.07 in most cases. Note that an offered traffic overload does not mean the system is in a congested state, since the allocation/admission control mechanism ensures that the system is not accepting more that it can handle (connections can be accepted with less than their maximum requirements).

![Utility performance for the hot-spots scenario](image)

Figure 5.4: Utility performance for the hot-spots scenario
Unbalanced load

In the first scenario we simulate the effects of a hot-spot area. We simulated a checkered pattern, where half of the cells are subject to a three times higher offered load than the others.

Compared to the hybrid-LP algorithm that represents the optimal allocation, and the pure-cellular which represents the optimal allocation without using ad hoc paths, the performance of the hybrid-heuristic is roughly in the middle, as shown in Figure 5.4. That is, the heuristic is around 10% better than the pure-cellular, and the LP is around 12% better than the heuristic. This is for moderate overloads such as 1.11 - 2.42. At heavy loads the “lighter” loaded cells are themselves quite overloaded and the algorithms tend to converge.

A more dramatic change can be observed in Figure 5.5. Here we plotted the utility generated by all connections originating from the MSs located in one of the heavy loaded cell. Being able to connect to the neighbouring cells, allows this set of MSs to generate 30% more utility with the hybrid-heuristic algorithm than in the pure-cellular setting. On the x-axis of the graph we have the half-hour simulated. The offered load in this setup is 2.42.

So, what explains the increase of service for the overload cell by 30% while the overall gain is only around 10%? This is because the higher importance connections that are accepted thanks to the ad hoc paths replace less important connections in the surrounding cells, so the absolute gain in utility is the difference between the accepted and the replaced. A direct effect of this load-balancing is that the degree of “QoS inversions” has been diminished. By “QoS inversions” we mean that connections/users with a higher importance are rejected in the overloaded cell while less important ones are serviced in the cells around.

BS failure and dead-spots

In the second scenario, Figure 5.6, we simulate the extreme case of a BS failure in one of four cells. The difference between the hybrid algorithms and the pure-cellular one increases greatly, as the pure
5.5. EXPERIMENTAL RESULTS

cellular network is clearly handicapped by not being able to use alternative paths to the direct cellular links. A similar situation arises in the handling of dead-spots. Dead spots are areas that are not covered by a BS due to obstacles or interference or big distances. The cell of a crashed BS becomes uncovered area, a big dead-spot. Therefore, these results should be proportionally applicable to dead-spot situations.

**Balanced load**

The third scenario presents a balanced load, which means that the MSs located in each cell generate on average the same amount of traffic. This is the scenario in which the hybrid network behaves most similar to a pure cellular network. When the load and type of the offered traffic is equally distributed, there should be no gain in sending connections to other BSs, especially considering the cost over the ad hoc paths. Nevertheless, at $1.11$ ($111\%$) offered

Figure 5.5: Utility originating in an overloaded cell
load the LP can take advantage of the global knowledge of load and ad hoc paths and shows a 10% improvement over the pure cellular setting, see Figure 5.7. The hybrid-heuristic relays only on local knowledge to choose the target BS, and acts close to the pure cellular, which is what we expected from such a scenario. As an overall trend we can observe that the highest benefits of using the ad hoc paths are gained for light to moderate overload. At underload and very high loads, the algorithms tend to converge.

5.5.2 QoS preservation

So far we have presented the results only from the perspective of the total system utility. Utility is also a good measure for how the QoS of the connections is respected, since each “QoS breach” is penalised (e.g. drop penalty). A more detailed view for the hybrid-heuristic algorithm at load of 2.42 is presented in Table 5.1. The application groups refer to those in Table 3.1. We can ob-
Figure 5.7: Utility performance for the balanced load scenario

serve that only connections that have the lowest utility efficiency are blocked (new connections) or dropped (ongoing connections). The allocation algorithms do not treat “ongoing connections” and “handovers” differently, both have a drop penalty attached, so if necessary, the connection with the lowest efficiency is dropped first. Since application group 6 is a class III connection, it can accept zero allocation situations, so there are no ongoing connections dropped in that case.

Table 5.1: Statistics per application group at load 2.42

<table>
<thead>
<tr>
<th>application group</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>accepted new</td>
<td>202</td>
<td>191</td>
<td>162</td>
<td>215</td>
<td>216</td>
<td>194</td>
</tr>
<tr>
<td>rejected new</td>
<td>0</td>
<td>0</td>
<td>58</td>
<td>0</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>rejected ongoing</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5.2: Computational complexity of the algorithms

<table>
<thead>
<tr>
<th>offered load</th>
<th>0.53</th>
<th>1.11</th>
<th>2.42</th>
<th>3.85</th>
<th>6.02</th>
</tr>
</thead>
<tbody>
<tr>
<td>hybrid heuristic</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>pure cellular</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>hybrid-LP</td>
<td>92</td>
<td>475</td>
<td>1196</td>
<td>2619</td>
<td>6894</td>
</tr>
</tbody>
</table>

### 5.5.3 Time complexity

Table 5.2 presents the running time (in seconds) of the simulator for different allocation algorithms, at increasing traffic loads. While there is no difference between the hybrid-heuristic and the pure cellular algorithm, the centralised linear programming algorithm is increasingly slower (30 – 200 times) and definitely not suited for an online allocation.

![Figure 5.8: Dependency on the cost parameter](image-url)
5.5.4 Cost influence

Using the ad hoc paths means using the equipment of other users in the area, and this translates further into a system-wide cost/hop/bit, $c$ that reduces the gained utility. Until now, we used $c = 0.02$, that is 20% of the efficiency of application group 3. Now depending on how big we assume this cost constant to be, the usage of ad hoc paths will be more or less encouraged. In Figure 5.8 we plot the dependency of the hybrid-heuristic algorithm on the cost/hop/bit. The baseline is the pure-cellular allocation. At zero cost, the overlay heuristic has a 20% advantage, however, as cost increases the advantage diminishes, and is on par with the pure-cellular at cost 0.1. This is a very high per-hop cost, since in our traffic mix, the efficiency of application group 3 is 0.1. That is, if we send it over a 1-hop path with cost 0.1, no utility would be gained. Thus we can conclude that for reasonable costs the hybrid-heuristic performs as desired. For very high costs (0.15 − 0.18) the “cost part” of the “path choice algorithm” becomes dominant and the algorithm uses only direct cellular links.

5.6 Summary

To satisfy increasing diverse access requirements and to improve availability, flexibility and higher data rates, today’s cellular networks can be foreseen to be replaced by hybrid cellular and ad hoc solutions. Moreover, services and applications with different QoS requirements will compete for the resources of such a network.

We showed that under reasonable assumptions, the bandwidth allocation problem can be formulated as a linear programming maximisation problem and thus optimally solved. While delivering optimal allocation, the running costs of the LP algorithm make it unsuitable for online allocations. Therefore, we proposed a low-cost heuristic algorithm that sends the connection to the BS, where it has the highest chance to be accepted.

The experiments show that the hybrid-heuristic algorithm has at worse 12% lower performance than the optimal one. In scen-
arios dealing with uneven traffic overload or coverage failure, the algorithm showed that it can take advantage of the hybrid setting and performed consistently better when compared to a pure cellular allocation (TARA). As opposed to the LP algorithm, the hybrid heuristic algorithm works in a distributed manner, being several orders of magnitude faster at runtime.
Chapter 6

Price-based distributed resource allocation

In this chapter we look at a more complex allocation problem than in Chapters 4 and 5, namely distributed bandwidth allocation in ad hoc networks. Due to the spatial reusability property, the wireless channel represents several “resources”, so the problem is a multi-resource allocation problem. Nevertheless, by using utility functions and their convex hull approximations, we can first formulate the allocation problem as a linear programming (LP) problem. The extended utility model of TARA is used, where the scheme aims to maximise the aggregated utility of the flows. As an abstract notion of resource we use maximal cliques of mutual interfering links.

In this chapter we propose a distributed algorithm that allocates the bandwidth based on flow-related bids for resources. These bids are calculated using the shadow price (from the dual LP formulation) of the resources, in combination with the flow’s utility function. The algorithm performs within 10% of a centralised LP allocation, that serves as an optimal though infeasible solution. In addition, we present a utility-aware on-demand “shortest” path routing algorithm in which the shadow prices are used as a natural distance metric.
6.1 Overview

Mobile ad hoc networks are formed by wireless nodes that move freely and have no fixed infrastructure. Each node in the network may act as a router for other nodes, and flows\(^1\) follow a multi-hop path from source to destination.

The infrastructure-less flexibility makes ad hoc networks a strong complement to cellular networks, suitable for many novel scenarios, including cooperative information sharing, defence applications and disaster management. Mobile ad hoc networks aim to provide a wide range of services, where best-effort, soft-real-time, and high priority critical data, should seamlessly integrate. As society becomes more dependent on such services, their availability under overload becomes a critical issue.

Ad hoc networks are more dynamic and unpredictable than their wireline or cellular counterparts. Mobility greatly influences the multi-hop routes, which are constantly created and destroyed (requiring flow rerouting in the latter case). Nodes that come too close to each other start interfere and cannot transmit independently. Thus we consider resource reallocation to be indispensable to both optimising and maintaining a feasible allocation. An ad hoc network designed for adaptive and autonomic reaction to failures and overloads, should take advantage of the flexibility in resource requirements of a supported service. Thus we use the extended TARA utility model, which ensures proper differentiation among flows and also proper consideration of requested resource guarantees.

In this chapter we present a combined routing, admission control and resource allocation scheme that aims to maximise the aggregated utility over flows and time. The wireless channel does not represent a single resource, due to its location-dependent reusability. Thus, we use the concept \textit{clique-resource} \cite{83, 84} that allows gathering mutually interfering links in partially overlapping maximal cliques. With the help of these cliques we can deter-

\(^1\)A flow represents a multi-hop, end-to-end connection. Throughout the chapter we use the terms interchangeably.
inistically account for bandwidth capacity and interference. A key concept in our distributed optimisation is the concept of resource shadow price, based on the dual of the linear programming formulation of the problem. Two novel, utility-based algorithms are presented. The first one is a distributed QoS-aware allocation algorithm. At each clique-resource on the end-to-end path of the flow, bandwidth is allocated independently, using only local information. Secondly, we present a complementary routing algorithm for choosing the most advantageous path for the flow.

We compare the performance of our distributed allocation algorithms with a “simple” baseline allocation scheme, that has no knowledge of utility. The “simple” algorithm expresses a traditional view on resource allocation, acting as an extension to the routing algorithm. Thus, finding a hop-based shortest path first (SPF) route is followed by reserving and allocating maximum possible bandwidth along the chosen route.

For an upper bound to allocation optimisation we provide an LP solution. While optimal, such an approach is infeasible for online allocation in ad hoc networks, since it needs a specialised node with powerful processing capabilities to solve the problem and global knowledge of the network.

Network-wide utility is our main performance metric, and we study the effects of mobility by varying the speeds of the nodes.

6.2 Related work

Work in resource allocation for ad hoc wireless networks has been addressed either at the MAC-level, as an extension to routing, or at an optimisation policy level.

For ad hoc networks bandwidth availability can be either pre-computed [83, 84, 85] or measured at MAC level [83]. Xue and Ganz [85] compute the available bandwidth at a node as the channel bandwidth minus the bandwidth consumed by the traffic at all neighbours. While easy to implement, this is too pessimistic, and better models can be created when interference structures are built.
based on link interference [83, 84]. In this work we use the contention model based on maximal cliques of contending links [83].

If no global optimisation is sought, resource allocation can be attempted independently at every node by appropriate MAC layer design. Luo et al. [84] present a packet scheduling approach to ensure a minimum weighted-fair scheduling combined with maximising spatial reuse of channel.

Routing is an important component in multi-hop wireless networks, and resource allocation/reservation is has been treated as an extension of the routing protocol. Karaki et al. provide a nice survey on QoS routing problems [89]. For instance Chen et al. [86] propose an on-demand distributed routing algorithm that aims to avoid flooding the network. They consider delay-constrained least cost and bandwidth constrained least cost problems. The work was extended by Liao et al. [87] to include the possibility to route a flow over multiple paths, increasing the chances of finding a route in very constrained systems. Routing algorithms might use also hierarchical schemes in order to alleviate complexity and overhead. The feature of the “bandwidth routing” [88] protocol is that link-layer scheduling is directly considered in the protocol. To calculate available bandwidth, both free bandwidth must be known, and a schedule of the free TDMA slots must be constructed.

QoS routing however, is usually not directly aimed at optimal resource allocation, but at finding the shortest path that satisfies at least some minimum QoS requirements, or the path that gives the largest margins for a QoS constraint. In this work however, the routing algorithm is part of the global allocation optimisation scheme.

Xue et al. [48], address a problem that is very similar to ours. They too use concave utility functions that represent user’s utility and aim to maximise the aggregated utility of the flows in the network. Both solutions use “shadow prices” of bandwidth resources on the end-to-end path of the flow for steering allocation. However, there are some fundamental differences between the two approaches. Xue et al. use non-linear functions while we use piecewise linear ones. Their work builds upon a previous problem
formulation by Kelly [47], for wireline networks. The allocation problem is split in two, where a) the network adapts to the rate of flows by changing resource price, b) the flow adapts to the new price by modifying the transmission rate. The iterative algorithm continuously changes the flow’s allocation until it reaches an optimal value, and this requires several hundreds of iterations. While this works for long-life flexible flows, continuous allocation changes are unacceptable for inflexible flows needing resource guarantees. Also, due to mobility and new flows arrivals/old flows departure, the system might spend little time in an optimal state, being forced to often restart the iterative adaptation process pursuing a new optimal allocation.

In our allocation scheme, reallocation is performed periodically, and thus, an allocation is kept unchanged for at least one period. At every (re)allocation decision point we aim to allocate close to optimal in a single try, and consider also the effects of a reallocation on the flow’s accrued utility.

In an extension, Xue et al. [83] use AODV [80] as a routing algorithm in a mobile environment. AODV routes over the shortest path (number of hops) and this might overload inner network paths while resources go unused towards the marginal areas. Therefore we use the price-based approach also for routing decisions.

6.3 Problem formulation

This section first lays out the network model, followed by a LP formulation of the allocation problem. Then, we present the notion of shadow price and some properties of the optimal solution, which will be used when constructing the distributed algorithm.

6.3.1 Network model

We consider a wireless ad hoc network with \( n \) nodes. Nodes \( a \) and \( b \) that are in transmission range of each other are connected by a wireless link, \( l_{ab} \). Nodes communicate with each other by
means of multi-hop bidirectional end-to-end flows \( (f_i) \) between an originator (source) node and a destination node.

In ad hoc wireless networks, we have a location-dependent contention between the transmissions on the wireless links. Transmissions over a link can be bidirectional, thus two links contend with each other if one of the end-nodes of a link is within the transmission range of an end-node of the other link [48, 84]. A link contention graph can be constructed, where vertices represent wireless links. In this graph, an edge connects two vertices if the corresponding links contend with each other. Each maximal clique\(^2\) in such a graph represents a distinct maximal set of mutual contending links.

A necessary condition for a feasible bandwidth allocation is that for each maximal clique the bandwidth used by all the links forming the clique is less than or equal to channel capacity. The channel capacity gives only an upper bound, as in practice, the choice of transmission scheduling algorithm, and even the topology of the cliques [90] can impose a tighter bound. That is,

\[
\forall j, \quad \sum_{l \in r_j} lb_l \leq B_j^{\text{max}}
\]

(6.1)

where \( lb_l \) is the allocated bandwidth over wireless link \( l \), \( r_j \) is a given maximal clique, and \( B_j^{\text{max}} \) is the achievable clique capacity, that is less or equal to the wireless channel capacity.

Hence, each maximal clique can be regarded as a distinct clique-resource with capacity \( B_j^{\text{max}} \). This is a fundamental different resource model compared to wireline networks. In wireline networks the capacity of a link is independent of the capacity of any other link, i.e. resources are formed by a single link. Only wireless links close to each other contend for the same bandwidth, so local information\(^3\) is sufficient for constructing the cliques that

\(^2\)A maximal clique is a subset of vertices, each pair of which defines an edge, that cannot be enlarged by adding any additional vertex.

\(^3\)We assume that the communication range is the same as the transmission range. Otherwise, bandwidth estimation has to be used, since two nodes could interfere but not be able to communicate.
6.3. PROBLEM FORMULATION

a certain link belongs to (details in Section 6.4.5). More description and proofs are given by Xue et al [48, 83].

In Figure 6.1 we present an example of a network topology (the mobile nodes are represented as squares) and two ongoing flows using this network. Figure 6.2 presents the link contention graph, where vertices represent the links (identified by corresponding numbers) of the network in Figure 6.1. We can identify three maximal cliques representing resources. Note that a single flow can span over several links belonging to the same clique-resource.

Let $q_{ij}$ represent the number of links of clique $r_j$ that are used by flow $f_i$. Transmissions over the links in a clique are mutually exclusive, and this means that the “effective” bandwidth used by the flow is $q_{ij}$ times higher than its nominal rate. Let $m$ be the total number of flows, and $b_i$ a certain allocation to flow $f_i$. Finally we can rewrite the constraints in Equation 6.1 in relation to the bandwidth (rate) allocated to the flows:

$$\forall j, \sum_{i=0}^{m} q_{ij} \times b_i \leq B_{j}^{max}$$ (6.2)

Table 6.1 presents the values of $q_{ij}$ for the example in Figures 6.1 and 6.2.
6.3.2 Linear programming formulation

For the wireless multi-hop network, having computed all the clique-resources, and assuming for now that for every flow the path between source and destination is set (routing done), at any allocation moment we can formulate the following problem. Let $u_i$ be the utility function and $x_i$ the allocation to be determined for flow $i$. Let $p$ be the number of clique-resources and $q_{ij}$ the usage counter of clique-resource $j$ by flow $i$. Then the optimal allocation
for all $x_i$ over all cliques $j$ can be obtained from:

\[
\text{Maximise } \sum_{i=1}^{m} u_i(x_i) \tag{6.3}
\]

subject to \[
\sum_{i=1}^{m} q_{ij} \times x_i \leq B_{j}^{\max} \tag{6.4}
\]

\[
x_i \geq 0 \tag{6.5}
\]

where $B_{j}^{\max}$ is the maximum bandwidth available for clique $j$.

\[\text{Figure 6.3: Linear segments of a convex hull}\]

In the next paragraphs, up to Lemma 6.3.1 we present the transformation of the problem above to a LP form, by using the same steps as in Section 5.3.3.

We first approximate the original utility functions, $u_i$, with their convex hull frontiers, $u'_i$ which are piece-wise linear and concave. To completely linearise the objective function we conceptually split a flow in several parallel subflows (same source, destination, and path), each corresponding to a linear segment of the utility function. For a subflow $k$ of flow $i$ the allocation is constrained as follows, $b_{i}^{k} \leq B_{i}^{k} - B_{i}^{k-1}$. The utility efficiency of the subflow (utility/bit) is $\lambda_{i}^{k} = \frac{U_{i}^{k} - U_{i}^{k-1}}{B_{i}^{k} - B_{i}^{k-1}}$. In Figure 6.3 we have an example of a convex hull with 3 linear segments corresponding to 3 subflows. Then given $s$ segments in the convex hull, for allocations
to subflows of flow $i$, we have:

$$u'_i(b_i) = \sum_{k=1}^{s} \lambda_k^i \times b_k^i$$

(6.6)

where $b_i = \sum_{k=1}^{s} b_k^i$. However, not every allocation to the subflows is consistent. In order to use the right side of Equation 6.6 as a function, we have to add two constraints:

$(C_1)$ Every $k$-th subflow has a maximum allocation limit, that is any $b_k^i \leq b_k^i \text{max}$ where $b_k^i \text{max} = B_k^i - B_{k-1}^i$.

$(C_2)$ The order of the segments in the R-U function must be respected when allocating (i.e. “gaps” are not allowed). That is, if $b_k^i > 0$ then for all $l < k$, $b_l^i = b_l^i \text{max}$.

Constraint $(C_1)$ is linear and can be directly used in the LP problem formulation, presented next. The subflow allocation variables are denoted by $x_k^i$.

Maximise

$$\sum_{i=1,k=1}^{m,s} \lambda_i^k \times x_i^k$$

subject to

$$\sum_{i=1,k=1}^{m,s} q_{ij} \times x_i^k \leq B_j^i \text{max}$$

$$0 \leq x_i^k \leq b_i^k \text{max}$$

Due to concavity, the linear segments of the convex hull are ordered highest efficiency first, and therefore an optimal allocation to the subflows automatically respects $(C_2)$, as proven by the following lemma.

Lemma 6.3.1 The results of the maximisation problem in Equations (6.7) to (6.9) satisfy constraint $(C_2)$.

Proof Let’s assume the opposite, which means that there are two subflows, $l$ and $k$, $l < k$, of a flow $i$ where $b_l^i < b_l^i \text{max}$ and $b_k^i > 0$. Let $\gamma = \min(b_l^i \text{max} - b_l^i, b_k^i)$. We denote the utility generated
by subflows $l$ and $k$ by $U_{l+k} = \lambda_l \times b_l + \lambda_k \times b_k$. Since both subflows belong to the same flow, one can imagine subtracting $\gamma$ from subflow $k$ and to allocate it to subflow $l$. Let $U'_{l+k} = \lambda_l \times (b_l + \gamma) - \lambda_k \times (b_k - \gamma)$. Then $U'_{l+k} > U_{l+k}$ because $\lambda_l > \lambda_k$ for a concave function. The allocations for other subflows being equal, this means that $b_l, b_k$ are not optimal. Contradiction.

Next we give a theoretical worst case difference between the optimal solution to the original maximisation problem (Equations (6.3) to 6.5), and the solution given by the above LP formulation that uses convex hull approximations when deciding allocation. Moreover, in Table 6.4 we present also a worst case difference computed at runtime during the allocation rounds.

Similarly to the single resource allocation bound in Section 3.5, let $\delta_i$ denote the maximum difference between the convex hull of the utility function and the utility function itself, for any of the flows involved. Let $\xi$ be the highest $\delta_i$ among all the utility functions of the involved flows, $\xi = \max_{i=1}^n \delta_i$.

Let $b_k^i$ be an allocation as obtained by solving the LP problem of Equations (6.7) to (6.9), $p$ be the number of clique-resources, and $U$ be the system-wide utility obtained by the LP allocation, i.e. $U = \sum_i u_i(b_i) = \sum_i u_i(\sum_k b_k^i)$. Finally let $U_{opt}$ be the optimal system-wide utility obtainable.

**Lemma 6.3.2** $U_{opt} - \xi \times p \leq U \leq U_{opt}$.

**Proof** Let $U_{aug}$ be the optimal value of the objective function of the LP problem, $U_{aug} = u'_i(b_i)$. Since the convex hull is larger than the original ($u'_i > u_i$) for all instances, $U_{opt} \leq U_{aug}$. Now, let’s see how large the difference between $U$ and $U_{opt}$ can become. Regarding the allocation of subflows, $b_k^i$, if $b_k^i = 0$ or $b_k^i = b_k^i_{max}$ the subflow has the same contribution to $U$ and $U_{aug}$. So the difference between $U$ and $U_{aug}$ is given by the number of subflows with partial allocation, i.e. where $0 < b_k^i < b_k^i_{max}$. These partial allocations can result only due to the constraints in Equation (6.8). For each of the $p$ constraints in Equation (6.8), at most one segment might result in a partial allocation. Thus $U_{aug} - U \leq \xi \times p$, which leads to $U_{opt} - \xi \times p \leq U \leq U_{opt}$. 

Unfortunately, the centralised allocation, and relatively large computational and signalling overheads make the LP solution infeasible for online allocation in an open, dynamic ad hoc network. Therefore we propose a distributed, low complexity allocation scheme, while using the LP allocation as an “upper bound” baseline to evaluate the performance of the distributed algorithm. For a distributed allocation algorithm we use the shadow price concept so we first present the dual formulation of the above LP problem together with some useful characteristics of an optimal solution.

6.3.3 Dual form and optimal solution properties

In economic terms the dual optimisation problem can be explained as follows. Assume somebody would like to buy a very small amount of a resource. The shadow (or marginal) price should be the minimum price the owner should accept. Obviously for offers lower than the shadow price the owner would gain more using the resource in his own production. Thus, the shadow price is a measure of resource contention and shows the marginal increase in utility if more resource would be available. The following is the dual of the previous LP problem in Equations (6.7) to (6.9):

\[
\begin{align*}
\text{Minimise} & \quad \sum_{j=1}^{p} B_j^{\text{max}} \times y_j + \sum_{i=1,k=1}^{m,s} b_i^{k,\text{max}} \times \nu_i^k \\
\text{subject to} & \quad \sum_{j=1}^{p} q_{ij} \times y_j + \nu_i^k \geq \lambda_i^k \\
& \quad 0 \leq y_j, \ 0 \leq \nu_i^k
\end{align*}
\] (6.10)  

(6.11)  

(6.12)

The shadow price of clique-resource \( j \) is denoted as \( y_j \), the number of all clique-resource being \( p \). Note that the shadow price is the price per unit of the resource (i.e. price/bit). Since subflows are by default constrained to a maximum bandwidth \( b_i^{k,\text{max}} \) this is modelled as a limited resource, which is used only by the respective
subflow, and \( v^k_i \) denotes the shadow price of this “artificial” limited resource.

We now define the slack variable \( w_j \) as the amount of unused capacity of clique-resource \( j \). For a given flow \( f_i \) the slack variable \( z^k_i \) represents “loss per unit”, i.e. the difference between the “benefit per bit”, \( \lambda^k_i \), and the sum of the shadow prices of used clique-resources. Using the slack variables, the inequalities of the primal and dual problem become:

\[
\begin{align*}
\sum_{i=1,k=1}^{m,s} q_{ij} \times x^k_i + w_j &= B_{j}^{\text{max}} \\
\sum_{j=1}^{p} q_{ij} \times y_j + v^k_i - z^k_i &= \lambda^k_i
\end{align*}
\] (6.13) (6.14)

According to LP theory, the optimal solutions for the primal and dual problems fulfil the following constraints [91] (Constraint (6.17) is similar to (6.16), but applies the above mentioned “artificial” resources):

\[
\begin{align*}
x^k_i \times z^k_i &= 0 \\
y_j \times w_j &= 0 \\
v^k_i \times (b_{i}^{\text{max}} - x^k_i) &= 0
\end{align*}
\] (6.15) (6.16) (6.17)

From equations 6.13-6.17 we can identify the following characteristics of the optimal solution:

\( O_1 \) If a resource is underutilised (\( w_j > 0 \)) then its shadow price is zero (\( y_j = 0 \)), otherwise its price is greater than zero.

\( O_2 \) For subflows where \( z^k_i > 0 \), we have \( \sum_j q_{ij} \times y_j > \lambda^k_i \), meaning that the accumulated price is higher than the subflow utility efficiency. Thus \( x^k_i = 0 \), and so \( v^k_i = 0 \).

\( O_3 \) For subflows where \( z^k_i = 0 \), and \( v^k_i = 0 \), we have \( \sum_j q_{ij} \times y_j = \lambda^k_i \). This means that the subflow is at the allocation edge given the resources it uses.
(O₄) For subflows where \( v^k_i > 0 \), we have \( \sum_j q_{ij} \times y_j < \lambda^k_i \). Also, \( x^k_i = b^{k \text{ max}}_i \) and \( z^k_i = 0 \). Parameter \( v^k_i > 0 \) represents a “pricing slack”, i.e. the amount by which the accumulated prices of the used resources could increase, and the flow still be profitable.

### 6.4 Distributed routing and (re)allocation

The ad hoc network considered in this work is an open dynamic system where resource requests and availability are always changing. Thus, our scheme employs periodic reallocations to keep the resource usage optimised. As end-to-end connections span several nodes and clique-resources, it is important that (re)allocations are well coordinated along the path. Also, reallocations imply a “mode” change for applications so their number should be strictly controlled. The use of periodic, synchronised allocation rounds guarantees that flows enjoy an allocation for at least one period. It also puts a bound on the reallocation rate in the system, even if the rate of events (traffic and topology changes) is much higher. The only disadvantage is that new and rerouted flows must wait until the next allocation time to receive new resources. Choosing an appropriate period implies tradeoff between a) utility optimisation and reducing the delay of path establishment and b) the computation and signalling overhead of allocation rounds.

The algorithm will be referred as adhoc-TARA in the rest of the chapter. Assume that a route for a flow is given (we will come back to how this route is found shortly). Conceptually, at each period the (re)allocation proceeds like this:

- Every flow calculates a bid for all clique-resources it traverses, based on their associated shadow prices.

- Each clique-resource independently evaluates the bids, proposes a certain bandwidth allocation to the flows and recalculates its shadow price.
The flow chooses the lowest bandwidth proposal from all the cliques it traverses as the new bandwidth for the new period.

Since a flow is constrained by the lowest available bandwidth on its path, the allocations must be performed synchronised at all clique-resources.

### 6.4.1 Bid construction

For each flow, the shadow prices (determined at the previous allocation round) of all clique-resources on its end-to-end path are used to calculate its path-price, \( pp_i = \sum_j q_{ij} \times y_j \). Intuitively, the utility efficiency, \( \lambda_i^k \), represents the maximum “budget” that could be “paid” to the traversed resources keeping the flow still “profitable”. Note that both \( \lambda_i^k \) and \( pp_i \) measure utility and price per bit. Thus we can now compute the price slack of every subflow as the difference between the utility efficiency and the path price, \( slk_i^k = \lambda_i^k - pp_i \). During a new allocation, the price could increase for any of the used resources, so we uniformly divide the slack among all the cliques it traverses (\( cc_i \) is the clique counter for flow \( i \), \( cc_i = \sum_j q_{ij} \)). Thus, for each clique-resource we create the following bid:

\[
\text{bid}_{ij}^k = y_j + \frac{\lambda_i^k - pp_i}{cc_i} = y_j + \frac{\lambda_i^k - \sum_j q_{ij} \times y_j}{\sum_j q_{ij}}
\]  

where \( \text{bid}_{ij}^k \) is the bid of flow \( i \), to resource \( j \) for subflow \( k \). The sum of a subflow’s bids amounts to its maximum “budget”, \( \lambda_i^k \). Thus, if all bids of a subflow are accepted, the subflow is accepted, and corresponds to either category \( (O_3) \) or \( (O_4) \) in Section 6.3.3.

### 6.4.2 Independent allocation

After all the bids have been placed, every clique-resource will independently allocate the bandwidth to the subflows, in decreasing order of bids, until bandwidth is depleted. Then the new shadow price of the resource is set to the price of the lowest bid among
the accepted subflows. Note that all the bandwidth is reallocated, and some subflows might get this time an allocation different from last period.

Then we have the following cases. If the contention at a certain resource is greater than at the previous allocation, its price will increase. If the bid of a subflow does not accommodate this increase, the subflow will be rejected. If the contention decreases, the price of the resource will decrease. This means that, some subflows that bid less than the previous price (i.e. have a negative price slack) are accepted, bringing the price down accordingly. If the resource does not allocate all bandwidth, it becomes underloaded and its shadow price becomes zero (case \(O_1\) in Section 6.3.3).

### 6.4.3 Discussions

Note that if we could use the solutions of the dual problem when constructing bids, any subflow, at all the cliques that it traverses, would consistently be accepted or rejected. As we do not know the new shadow price beforehand, we use the shadow price from the last allocation as an estimate. If at a clique-resource the contention level has increased, the price could increase more than the bid of some subflows (resource price was underestimated at bid construction). In such a case, some subflows that could have offered a proper bid (with hindsight) are rejected. Conversely, overestimating a resource price unnecessarily, increases its bid to the detriment of others.

As a consequence of over/underestimation, for a flow, bandwidth could be allocated in different amounts at different clique-resources, and the flow can use only the minimum allocation over the end-to-end path. As a remark, we want to mention that in such cases one might use the algorithm iteratively, to better balance the bids. Nevertheless, as the algorithm is intended for online allocation, we do not iterate and any mis-allocated bandwidth remains unused. Since in an optimal allocation, the amount of this mis-allocated bandwidth would be zero, we will use it in our experiments as another measure of how close to the optimal allocation our algorithm performs.
6.4. DISTRIBUTED ROUTING AND ALLOCATION

Allocation algorithm run at every clique-leader $j$, at every period $T$:

Let $F_j$ be the set of flows using resource $j$

\[ awb_j = B_j^{\text{max}} \]  

// initialise available bandwidth

\[ \forall \text{ subflows } f^k_i \in F_j \]

\[ bid^k_{ij} = y_j + (\lambda^k_i - pp_i) / cc_i \]  

// compute bid

while $F_j \neq \emptyset$  

// allocate for highest bidder first:

select $f^k_i \in F_j$ with highest bid

if $awb_j > q^k_{ij} \times b_i^{k, \text{max}}$

\[ x_{ij}^k = b_i^{k, \text{max}} \]

\[ awb_j = awb_j - q^k_{ij} \times x_{ij}^k \]

else

\[ x_{ij}^k = 0 \]

$F_j = F_j - f^k_i$

\[ y_j = \min(\{bid^k_{ij} \mid x_{ij}^k > 0\}) \]  

// recompute resource price

\[ \forall i \text{ where } f_i \in F_j \]

\[ x_{ij} = \sum_k x_{ij}^k \]

send $x_{ij}$ and price $y_j$ to source of $f_i$

Flow adaptation algorithm run at every node $n$, at every period $T$:

\[ \forall \text{ flows } f_i \text{ sourced at node } n \]

\[ \forall \text{ resources } j \text{ that } f_i \text{ traverses} \]

wait for allocation $x_{ij}$ and price $y_j$

\[ x_i = \min(x_{ij}) \]  

// set bandwidth of $f_i$

\[ pp_i = \sum_j q_{ij} \times y_j \]  

// recompute its path price

\[ \forall \text{ resources } j \text{ that } f_i \text{ traverses} \]

send $pp_i$ to clique-leader of resource $j$

Figure 6.4: The distributed allocation algorithm
Figure 6.4 presents a pseudocode of the distributed algorithm that is run synchronised for every clique-resource and respectively for every flow. For every clique-resource, a clique-leader node, which is used for performing the (re)allocation computations, is determined at clique-construction time, see Section 6.4.5. Whenever a flow starts/stops using a wireless link, one of the end-nodes of the link registers/deregisters the flow (i.e. its source node) with the clique-leaders of all the cliques containing that link. The clique-leader gathers information about the flows using the clique-resource and runs the allocation algorithm. The natural place for running the flow-part of the algorithm (and changing the transmission rate) is at the flow’s source node.

Note that the signalling information (between clique-resources and the flows’ source nodes) is sent only along established flows, and thus can be piggy-backed on existing packets. The clique-leader can be chosen such that the distance to the end-nodes of the links belonging to the clique is at most 2 hops (see also Section 6.4.5). Therefore, inside-clique signalling could use the MAC layer signalling (e.g. piggyback RTS,CTS,ACK packets). As reallocations occur seldom (compared to flow dynamics) we envision a small signalling overhead.

The flow’s source node must receive the new bandwidth decision from all the clique-resources on the end-to-end path of the flow and choose the minimum allocated. The larger the synchronisation error between the clocks of the clique-leaders, the more the source-node has to wait until it can set the new rate of the flow. As a consequence, a flow might increase its rate before another decreases it, leading to short-lived congestions at certain points. Regarding clock-synchronisation protocols in wireless (sensor) ad hoc networks, Römer et al. [92] give precision results of less than 100µsec for nodes five hops away. These clock skews are small compared to the envisioned reallocation periods, and thus we assume these congestions to be easily mitigated. Nevertheless, synchronised allocation generate bursty signalling in the network, and our ongoing work studies methods that potentially remove the need for synchronised allocation rounds among the clique-resources.
6.4.4 QoS routing

Traditional QoS routing algorithms typically use either shortest path (respecting minimum constraints), or widest path (allowing a better QoS for that flow). However, these are two extreme cases and do not optimise global utility. Shortest path might overload some routes. Widest path may produce too long routes, increasing total network load. Therefore, as part of adhoc-TARA we propose a new routing algorithm based on the shadow price of resources introduced above. Used with the allocation algorithm presented in Section 6.4.3 the best chance for the highest QoS is along a path with the lowest path price. So we use an on-demand shortest path first routing algorithm that uses the path price as distance metric (i.e. it chooses a path that yields a minimal $pp_i = \sum_j q_{ij} \times y_j$). The lowest path price comes from both less contended links (lower link prices) and shorter topological paths (lower number of links in the path).

Once chosen, keeping a route fixed is important for deterministic resource allocation. For this we use source-routing. In source routing the source specifies the hops to destination, and routing tables are not needed. All else being equal, source routing may have a higher overhead due to the bigger routing packets. However, table routing protocols have usually only one entry in the table per destination. Thus the possibility of using alternative paths by source routing algorithms provides an advantage in QoS-aware systems. Besides providing load balancing capabilities, it prevents load oscillations when e.g. the shortest path changes.

Rerouting is performed for flows only when a link in the end-to-end path breaks due to mobility. There are two reasons why not to perform rerouting in the case of a decrease in allocation. First, this would create an oscillating allocation pattern where flows constantly chase a better route. Second, rerouting implies a big signalling overhead, and should be used only if necessary. Routing or rerouting is asynchronous to allocation, however the shortest paths are valid only until next allocation round (due to price changes).
6.4.5 Mobility and clique construction

Due to mobility, a node might enter or exit the communication range of another one, thus creating a new wireless link, or alternatively breaking one. Handling topology changes can be implemented either event based (when MAC feedback is used) or periodically (when hello messages are broadcast), and should be independent of the allocation algorithm.

<table>
<thead>
<tr>
<th>Clique reconstruction algorithm at every node $n$, at every period $T'$:</th>
</tr>
</thead>
<tbody>
<tr>
<td>local (one-hop) broadcast “hello” message</td>
</tr>
<tr>
<td>timeout-wait</td>
</tr>
<tr>
<td>if the neighbour set changed</td>
</tr>
<tr>
<td>notify sources of broken flows to reroute</td>
</tr>
<tr>
<td>three-hop broadcast the new and broken links</td>
</tr>
<tr>
<td>timeout-wait</td>
</tr>
<tr>
<td>(re)construct clique-resources due to new and broken links</td>
</tr>
<tr>
<td>$\forall$ new resources $j$ where $n$ is clique-leader</td>
</tr>
<tr>
<td>establish initial $y_j$</td>
</tr>
<tr>
<td>$\forall$ flows $f_i$ traversing resource $j$</td>
</tr>
<tr>
<td>send $q_{ij}$ and $y_j$ to source of flow $f_i$</td>
</tr>
<tr>
<td>timeout-wait</td>
</tr>
<tr>
<td>$\forall$ flows $f_i$ sourced at node $n$</td>
</tr>
<tr>
<td>if $f_i$ is broken</td>
</tr>
<tr>
<td>reroute $f_i$</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>$pp_i = \sum_j q_{ij} \times y_j$ //recompute its path price</td>
</tr>
<tr>
<td>$cc_i = \sum_j q_{ij}$ //recompute its clique-counter</td>
</tr>
<tr>
<td>$\forall$ resources $j$ that $f_i$ traverses</td>
</tr>
<tr>
<td>send $pp_i$ and $cc_i$ to clique-leader of $j$</td>
</tr>
</tbody>
</table>

Figure 6.5: The clique reconstruction algorithm

For clique (re)computation, new and broken links should be reported to all nodes as far as 3 hops away. Thus any node in the network will have enough information to construct a link contention graph that includes all the wireless links contending with any of its adjacent links. This local topology knowledge is enough
for a node to identify all the maximal cliques that contain any of its adjacent links. To identify the cliques in the local network we have used the Bierstone algorithm [93, 94]. So all the nodes in the network identify the relevant clique-resources independently.

In Figure 6.5 the clique-reconstruction algorithm is presented. All the nodes identify all the relevant clique-resources independently. Then, for each clique-resource the “clique-leader” node is determined (e.g. the node in the clique that has the highest identifier, and is adjacent to at least two links belonging to the clique). The algorithm implies waiting for results from other nodes, and thus runs synchronised with a period $T'$.

If a link breaks, all the flows that used this link should be rerouted. Some old clique-resources will disappear and some new ones will be created. The new cliques will have an initial zero shadow price. However, to set a better starting price, we perform a “dry allocation” (no bandwidth is actually reallocated) at the new clique-resources, based on the inherited flows. After a topology change all the affected flows must update their path price, to be used in the next allocation round.

6.5 Simulation and results

6.5.1 Evaluation setup

To evaluate the behaviour of our resource allocation scheme we use a traffic mix, that is very similar to the one presented in Section 3.6 and used in Chapters 4 and 5. The difference is that we have slightly changed the bandwidth requirements of the application groups for their usage in an ad hoc network. Table 6.2 summarises their characteristics, with the utility functions unchanged from Section 3.6.

A simulator was built on top of the J-Sim component platform [61], however packet level simulation was not considered at this stage. The experiments use $1500 \times 1500m^2$ area where 60 mobile stations are uniformly, randomly deployed. The communication range is $250m$ and considered equal to the interference
Table 6.2: Traffic mix used in the ad hoc experiments

<table>
<thead>
<tr>
<th>Applic. Group</th>
<th>Max. Bandwidth Requirement (Kbps)</th>
<th>Connection Duration (sec)</th>
<th>Examples</th>
<th>Class</th>
<th>Utility Scaling Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>Max</td>
<td>avg</td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>60</td>
<td>600</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>256</td>
<td>128</td>
<td>60</td>
<td>1800</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>1000</td>
<td>500</td>
<td>300</td>
<td>7200</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>512</td>
<td>256</td>
<td>30</td>
<td>7200</td>
</tr>
<tr>
<td>6</td>
<td>128</td>
<td>2000</td>
<td>512</td>
<td>30</td>
<td>1200</td>
</tr>
</tbody>
</table>

range. Environmental perturbations are not considered, and every clique-resource has the 4Mb/s channel bandwidth at their disposition.

Mobility is implemented using the random way-point model, with a random speed between 0 and max speed. To ensure a nice connectivity and keep the mobile nodes from clumping together, we made nodes move away from each other when they come closer than a third of the communication distance. New flows arrive following an exponentially distributed inter-arrival time. All the 6 application groups arrive with equal probability. Figure 6.6 gives a snapshot of the ad hoc network topology, during one of the experiments.

6.5.2 Comparison of allocation schemes

In our experiments we compare the behaviour of the following routing and allocation schemes for different load and mobility scenarios.

- As a baseline algorithm we use a non-utility routing and allocation scheme denoted by simple in the experiments. The routing is on-demand shortest path first (hop-based). After a route is chosen, the maximum bandwidth that can
Figure 6.6: Ad hoc network snapshot

be provided by at all clique-resources on the end-to-end path is allocated to the flow. If not enough bandwidth is available to accommodate the minimum then the flow is rejected. If the path breaks, the flow is rerouted, and new bandwidth is allocated. If a clique-resource becomes overloaded due to mobility, flows will be dropped on a last-accepted first-rejected basis. We do not reroute in this case, since most likely the same overloaded path will be chosen by the SPF routing algorithm, and some connections have to be dropped in order to relieve congestion. On the other hand, a path may break regardless of the load in the network.

- To represent best possible solution, we use a LP solver to optimally solve the global allocation problem as defined in Section 6.3.2. The formulation of the LP problem does not include routing, so we use the price routing algorithm described in Section 6.4.4. This serves to compare our distributed allocation algorithm with the optimal allocation.

- Next we show the results of runs for the adhoc-TARA scheme. It uses the distributed allocation algorithm described in Sec-
tions 6.4.1 to 6.4.3 and the price-based routing algorithm from Section 6.4.4.

- Finally, we compare with a variant of our distributed allocation scheme, denoted \textit{altbid}, where a different formula is used to construct the bids. In this alternative the “budget”, $\lambda_i^k$, is proportionally divided based on shadow prices. Thus, 
\[
\text{bid}_{ij}^k = \frac{y_j \times \lambda_i^k}{\sum_j q_{ij} \times y_j}.
\]
The intuition is that resources with higher shadow prices are more disputed, and thus have a higher chance of getting even more disputed. However, bids for low priced resources become very small, and in the case of a 0 priced resource, all bid$_{ij}^k = 0$. In this case, ties are broken by 
\[
slk_i^k = \lambda_i^k - \sum_j q_{ij} \times y_j.
\]

For all the four schemes allocation is separated from utility accounting. Utility accounting is performed using the TARA utility model, presented in Section 4.4.

6.5.3 Experimental results

As utility is our main performance metric we first show how the total utility of the four schemes behaves when subjected to scenarios with different mobility. Thus in Figures 6.7 and 6.8, on the x-axis we have the average speed of the nodes (m/s), and on the y-axis the time-accumulated system utility. Every point represents an average of 3 different experiments. Each experiment was run over a time interval of 600sec, with a (re)allocation period of 2 seconds. All the experiments were exposed to a moderate offered traffic overload (average inter-arrival rate of 1/200s).

The experiments in the two figures are differentiated by the type of the applications used. In Figure 6.7 (“rt-mix” scenario) we consider a mix of rigid and flexible application groups as presented in Table 3.1 (see the “class” column). In Figure 6.8 we consider that all the 6 application groups are fully flexible (their class is set to class III). In this case, no flows will be dropped due to zero allocation.
We can see that the results of adhoc-TARA come surprisingly close to the optimal LP allocation. Even at the lowest point, the distributed allocation algorithm is at almost 90% of the optimal allocation (note the LP algorithm uses the same routing as adhoc-TARA). Both the “flexible” and the “rt-mix” scenarios suffer from mobility in similar ways (that is, the “rt-mix” scenario is not affected more). The simple scheme cannot differentiate properly between flows, and is trailing at around half of the utility of the LP algorithm. The altbid algorithm performs constantly below adhoc-TARA (at worst 72% of the LP). This is because the bid is too biased towards high-priced resources, while low priced resources can also quickly increase their prices. Adhoc-TARA, in comparison to altbid, creates a more evenly distributed bid.

In Table 6.3 we present the bandwidth utilisation of adhoc-TARA as compared to the utilisation of the LP algorithm. The
mobility row shows the average speed of nodes (m/s). Our distributed algorithm independently allocates bandwidth at the clique-resources along the flow’s path. If allocations are different, some bandwidth is wasted. The LP solution is using global knowledge, and thus has no such problem. Nevertheless, the difference in bandwidth usage (as an average over all clique-resources) between LP and adhoc-TARA is only around 15% as presented in Table 6.3.

Table 6.3: Bandwidth usage of adhoc-TARA compared to LP

<table>
<thead>
<tr>
<th>mobility</th>
<th>0</th>
<th>2</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>usage(%)</td>
<td>92</td>
<td>84</td>
<td>80</td>
<td>87</td>
</tr>
</tbody>
</table>

In Figures 6.7 and 6.8, we have presented the LP solution as optimal. Nevertheless, as shown by Lemma 6.3.2 there is a small
difference between the LP solution (that uses the convex hull approximation of the utility functions), and the true optimum. In the proof of Lemma 6.3.2, we have shown that $U \leq U_{opt} \leq U_{aug}$. As we can compute $U$ and $U_{aug}$ when running the allocation algorithm, this might yield a better difference bound between $U$ and $U_{opt}$. Let’s denote it as $\Delta U_{opt} = U_{aug} - U \geq U_{opt} - U$. Note that this bound only pertains to the accuracy of the LP formulation of the allocation algorithm, without applying TARA utility function modifications.

During an experimental run, 300 allocation rounds are performed during the simulated half hour. We varied the load in the network by increasing the inter-arrival rate of new flows. Table 6.4 shows the results of the experiments. We can observe that the optimal results are not more than 5% away from the LP allocation, even though the flows used in our experiments are quite large compared to the capacity of the wireless channel. Thus we expect a network populated with a large number of small flows to provide even better bounds.

<table>
<thead>
<tr>
<th>flow inter-arrival</th>
<th>1/400</th>
<th>1/200</th>
<th>1/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>avg. #conns</td>
<td>14</td>
<td>29</td>
<td>60</td>
</tr>
<tr>
<td>avg. $\Delta U_{opt} / \text{avg. } U$</td>
<td>0.043</td>
<td>0.047</td>
<td>0.040</td>
</tr>
</tbody>
</table>

A more detailed view on the QoS for the 6 application groups (from Table 3.1) is given by the experiments presented in Table 6.5. The two cases have the same offered load (the number of arriving flows is around 190), and are using adhoc-TARA scheme, and “rt-mix” traffic. We show the number of flows that were blocked and dropped, and also the average allocation (level) granted to a flow as a percentage of the maximum it requested. As expected, flows with lower general utility (groups 3 and 6, see last column in Table 3.1) fare worst. Also, the mobile case (second row) creates more congestion and mis-allocations than the static case (first row), and performance drops in general. If utility is high enough,
and mobility is not excessive, the QoS of rigid real-time flows is well preserved (e.g. application group 1).

Table 6.5: Statistics at application group level

<table>
<thead>
<tr>
<th>appGroup</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>mobility</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>mobility</td>
<td>0</td>
<td>1</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>level(%)</td>
<td>100</td>
<td>84</td>
<td>54</td>
<td>98</td>
<td>72</td>
<td>68</td>
</tr>
</tbody>
</table>

The next set of experiments, Figure 6.9, show how utility depends on the offered load. In all experiments average speed is 4. On the x-axis we plot the average inter-arrival rate for a light (1/400), moderate (1/200) and heavy (1/100) offered load. We can observe that the utilities of all 3 schemes follow similar trends and increase almost proportionally with load, preserving the superior performance of adhoc-TARA.

Besides signalling overhead, computational complexity is a big drawback of the LP solution and one of the reasons to choose adhoc-TARA. Table 6.6 gives a comparison of the average time (seconds, on a 1GHz P III) needed to reach an allocation decision, as offered load is increased.

Table 6.6: Average running time of allocation decision

<table>
<thead>
<tr>
<th>inter-arrival rate</th>
<th>1/400</th>
<th>1/200</th>
<th>1/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>54.1</td>
<td>173.8</td>
<td>890.9</td>
</tr>
<tr>
<td>adhoc-TARA</td>
<td>0.179</td>
<td>0.265</td>
<td>0.476</td>
</tr>
</tbody>
</table>

The above experiments gave a good indication of relative merits of the the distributed allocation algorithm of adhoc-TARA compared to optimal LP, as in both cases we use the price-based routing algorithm. Current work involves isolating the characteristics of the price-based routing. Surprisingly, preliminary results,
using networks with no hot-spots, show no important utility improvements by using the price-based routing instead of the hop-based shortest path first (SPF). However, more work needs to be done to study price-based routing in congested networks.

6.6 Summary

In this chapter we have presented a novel utility/price-based bandwidth allocation scheme for wireless networks, together with a complementary price-based routing algorithm.

We first show that we can use utility functions together with techniques derived from linear programming for optimising resource allocation in multi-hop ad hoc networks. We then propose a distributed allocation algorithm that bids for resources depending on their “shadow prices”, and the “utility efficiency” of the
flow. Simulations show a superior behaviour of the distributed allocation algorithm, which comes close to the optimal linear programming allocation, and has a much lower overhead.
Chapter 7

Dealing with Uncertainties

Previous chapters dealt with deterministic algorithms. These algorithms assume that the amount of required and available resources is well known, and that the periodic reallocation strategy can quickly react to topology and traffic changes.

However, as mentioned in Chapter 2, when allocating resources in open, dynamic systems such as wireless communication networks there are several uncertainties to be dealt with due to changing traffic patterns and mobility. Usually traffic generation is not known a priori, and mobility can swiftly change network topology. In addition to this, the resource requirements of applications and services can only be estimated, and may vary among different instances. For example, worst case execution time of tasks on a network controller could be several times higher than in the average case.

In this chapter we present strategies to deal with inherent processor load uncertainties in future generation mobile networks. We address the interplay between user differentiation and resource allocation, and specifically the problem of protecting the CPU of a radio network controller (RNC) from overloads.

Although processing capabilities in a wireless network are not usually regarded as a bottleneck resource (we do not know of other
In the presented algorithm we combine feedback, to control the load on the CPU, with share-based task differentiation, which ensure that the available load is partitioned corresponding to a certain policy. As opposed to earlier chapters that require the provision of user utility functions as a basic premise, the scheme in this chapter takes a pragmatic view that builds only on the input available in today’s systems.

From an architectural viewpoint, this work can be included in the category of low-level QoS enforcement mechanisms, compared to the high-level QoS management policies presented in the previous chapters.

7.1 Adaptive load control

The complexity of traffic models generated by advanced multimedia and data services, combined with user mobility and radio channel maintenance leads to the execution of various control functions in the radio network controller (RNC) in order to dynamically allocate data and control channels (types and rates), set up and tear down connections, and so forth.

The amount of control functions generated might vary greatly and might lead sometimes to the overload of the processor on which they execute. For example, a change in the bandwidth needs of a user triggers the execution of both bandwidth-related admission control functions and bandwidth allocation functions. Also, due to mobility, an unexpected amount of users may gather in the same place, increasing the load on the CPU responsible for that area.

Note that even when operating at maximum bandwidth allocation levels there will be new connection requests for the CPU, which have to be processed and rejected. Thus, even though the number of users accepted in the system is bounded by the available bandwidth, the load generated on the processor is not.
Therefore dealing with the more scarce resource (the bandwidth) will not eliminate CPU overloads. To solve this problem by over-provisioning would be wasteful since such overloads should not be a common work situation. If the processor is heavily overloaded then even the most important tasks (functions) can be indefinitely delayed and required response times will not be met.

The result of load control algorithm may be the rejection of some control functions, with adverse consequences on some ongoing applications and their respective users. Since different participants may have different resource requirements or importance, a differentiation scheme also needs to be in place.

7.2 Related work

Our load control scheme can be divided into two parts, a feedback-directed load control part and a differentiation mechanism. Related work on both these problems is presented next.

Differentiation mechanisms use either a share based or an absolute priority based approach when allocating a resource. The problem with absolute priorities is that they do not support quantitative allocation of resources. Thus in an open environment resources might be monopolised by ill-behaving, high priority applications.

A share-based allocation of the network link capacity is given by weighted fair queuing (WFQ) disciplines. An idealised model, using “fluid flows” (e.g. packet size is infinitesimal), is proposed by Parekh and Gallager as Generalised Processor Sharing (GPS) [35]. The basic idea behind GPS is the following. A weight $\phi$ is associated with each flow, and the link capacity is shared among the active flows in proportion to their weights. Therefore, if flow $i$ is active, it will be serviced with rate $\phi_i \times \gamma / \sum_{j \in A(t)} \phi_j$, where $\gamma$ is the link speed and $A(t)$ is the set of active flows. This means that the flow has a minimum guaranteed rate (if all flows are active) and that the scheduling discipline is work-conserving\(^1\). Moreover,

\[^1\text{A work-conserving scheduler is idle only if there is no traffic to send.}\]
if some flows are not active, the excess capacity is shared between the different flows in proportion to their respective weights.

Several variants of GPS have been proposed for packet scheduling networks, for example PGPS (Packet GPS) [35] or WF2Q (Worst Case Fair Weighted Fair Queuing) [95]. The technique has been extended to processor sharing by Stoica et al. [96], who proposed Earliest Eligible Virtual Deadline First (EEVDF), where the scheduling entity is not a packet but a quanta of the processor time.

To schedule CPU tasks closer to the needs of soft-real-time periodic tasks (multimedia), Tokuda et al. propose Processor Capacity Reserves [7], where a fraction of the CPU utilisation is reserved to each task. A reserve is a tuple \((C_i, T_i)\) indicating that a task can execute for at most \(C_i\) units during each period \(T_i\), the reserved utilisation share being \(\rho_i = C_i/T_i\). When a task exceeds the \(C_i\) allocated units, it is scheduled in the background until the new period begins. The system is dependent on real-time scheduling algorithms like rate monotonic (RM) or earliest deadline first (EDF) for scheduling. Admission control based on the schedulability test of the underlying scheduling algorithms (RM or EDF) is used when accepting the reservation requests of an application.

Abeni and Buttazzo propose the Constant Bandwidth Server (CBS) for temporal isolation between tasks and proportional sharing of the CPU inside an EDF environment [8, 9]. Each task which runs inside a CBS receives a budget \(Q_s\) to be spent with period \(T_s\). If the job executes for \(Q_s\) (time) units, the scheduling deadline is postponed by \(T_s\). Thus spare time is automatically and fairly reclaimed by the participating tasks. Admission control must be employed to keep the system schedulable.

All the previous techniques directly use a scheduler to enforce their share-based reservations. The difference in our case is that we do not assume any scheduling policy for the RNC. We only use an admission control strategy, which has to enforce the allocation policy regardless of the underlying operating system.

Recently, the area of automatic control, in which feedback is a means of dealing with uncertainties, has influenced the work
7.3 Load control in 3G networks

In this section a more systematic view of the load control possibilities is given. The RNC is the unit responsible for controlling the common resources in the 3G radio access network. The hardware in each RNC consists of a number of processors, each responsible for handling communications within a certain part of the network. A user request arrives at one of these CPUs to begin with. Several control functions are used for treating the request and managing the resulting connection if the request is accepted. The network should, however, have mechanisms to deal with unexpectedly high loads created by the control functions. The load is being primarily caused by traffic control functions, but also by functions on behalf of the higher network management layers and other processes generated by the operating system (OS) running on the RNC (traffic control functions were presented in more detail in Section 4.3).

Now, the purpose of the algorithm proposed in Section 7.4 is to accept as much load into the processor as possible without causing an overload on the CPU. This is because CPU overloads might be generated not only as a consequence of network load, but also due to errors or in the software/hardware environment. To avert
this situation a hardware based resetting mechanism is activated
during overloads (if the processor stays at 100% utilisation for a
certain period of time). As a consequence of resetting the CPU, all
the connections active in the network part for which this processor
is responsible will be disconnected. Moreover, the RNC becomes
unavailable during the time the system board is reinitialised.

To control the load on the processor we could either reject or
delay some of the above-mentioned control functions. A prelimi-
nary study of deferral methods shows that delaying traffic functions
is not a solution [56]. Moreover many of these functions should
not be subjected to rejections. Mobility management functions
have to be executed in order to maintain the connection status of
the mobile station as it moves across cells. Rejecting basic con-
trol functions like power control would disturb the existing con-
nections. Similarly, functions which perform disconnection from
the network should not be refused. Network operation and main-
tenance (O&M) functions are performed infrequently and usually
when the network is lightly loaded, so they are not good can-
didates either. This leaves us with three candidates: signalling
connection setup, radio access bearer (RAB) establishment and
channel switches. Since RAB establishment is usually performed
directly after a signalling connection setup, its control would not
bring any benefit. Therefore we concentrate on two traffic function
types: requests for new connections \(r_1\) and requests for channel
switches \(r_2\). Rejecting a new connection is deemed less penali-
sing than dropping an established call, therefore the former ought
to get a lower importance compared to the latter. All the other
traffic functions (tasks) together with other OS tasks are assumed
to be directly sent to the processor and outside our control (re-
ferred to as direct tasks).

Cast as a control problem in its simplest form, we have the
schematic diagram from Figure 7.1: the CPU to be protected from
overload is our "plant" in control terminology. The arriving re-
quests are inputs to the controller (CTRL), which has the primary
task of protecting the CPU from overloads. To do this it has ac-
cess to the actual measured CPU utilisation, which is the sum of
utilisation caused by the number of accepted requests (denoted by \(a\)) and the utilisation from direct tasks (denoted by \(U_d\)). Note that in this chapter \(u\) stands for utilisation, not utility. There are certain levels of desired utilisation, denoted here as a set value \(U_s\). The reason for having a \(U_s\) that is different from 100% is to have some spare processing power that allows the system to cope during transient states when admission control is not fully able to regulate the incoming load. Otherwise the hardware may reset itself and would jeopardise network availability during high load situations. Thus we want to keep the load on the CPU (denoted by \(U\)) to stay close to \(U_s\) (say 90%). Let \(C_i\) be the computation time for task type \(i\). In order to evaluate the utilisation caused by the mixture of arriving tasks \((r = r_1 + r_2)\), we represent the required utilisation for each task type by \(c_i\), where \(c_i = C_i/P\) and \(P\) is the sampling interval. Thus, \(U = U_d + \sum a_i c_i\), where \(a = a_1 + a_2\) are the accepted tasks made up of types 1 and 2 respectively (new connections or channel switches). To maintain \(U\) as close as possible to \(U_s\) we should calculate \(a_i\) from \(U_s = U_d + \sum a_i c_i\). However we cannot measure \(U_d\), and even the computation time \(C_i\) might be only an approximate value, therefore we feed back the resulted \(U\) to aid in the control of the admitted tasks \(a_i\).

So far we differentiate only between task types. Next we add a user dependent QoS dimension. That is, we represent different user (or application) classes by the index \(j\). The request type \(i\) from user type \(j\) is now represented by \(r_{ij}\). Then we can observe the network behaviour by how the mix of user types and the mix of task types changes over time.
7.4 Algorithm design

The above parameters make several uncertainties explicit. We distinguish between two types of uncertainty: the resource needs of arriving requests and the variation over time with respect to user service policies.

- Different user sessions (connections) lead to different volumes of channel switches that in turn generate different volumes of tasks to be executed on the CPU (depending on whether the user is using voice, mail, SMS, web, etc.). In this category of uncertainty we also find the volume of direct tasks arriving at the CPU ($U_d$ above) and the claim on the processor with respect to each task ($c_{ij}$).

- The second type of uncertainty, namely the differentiation policy for user/task types, is treated at a higher level. It can be affected by the market and pricing information, by configuration changes known at higher layers of network management, and by changes in the load generated by a particular user/task type.

At the RNC level, we use variants of automatic control techniques to adjust for the first type, and a deterministic algorithm to reflect the policy to deal with the second type. A change in the policy can thus be seen as the adjustment in the service level delivered to a particular user class. For instance, the differentiation strategy could result directly from a bandwidth allocation algorithm such as the one presented in Chapter 4. We could allocate the CPU based on absolute priorities. In this case however, the control tasks of a higher priority connection could monopolise the CPU power. That is, it could lead to the rejection of several connections that have low CPU demands, but together generate more utility than the misbehaving task. The option we chose to implement is to use quotas (e.g. proportional to the bandwidth allocation). These should constrain misbehaving connections and protect the rest. We also like to expose some work-conserving abilities. That is, if there is not enough load generated by a of
a certain connection type, the remaining of its quota should be reassigned to the overloaded types proportionally.

**Proposed architecture**

Figure 7.2 illustrates our architecture for the first approach, where the Load Control Unit (LCU) is instantiated by two boxes dealing with the two types of uncertainties mentioned above. The load quotas allocated to the different task categories are established at the allocation policy level as discussed above. Though represented as a (leftmost) box in our architecture, it is not part of the control algorithm presented in this chapter.

![Figure 7.2: Schematic description of the architecture](image)

- $r_{ij}$ - is the number of incoming user requests of a given task type-$i$ from user type-$j$.
- $p_{ij}$ - specifies the percentage of the available utilisation which should be allocated to user requests of type $(i, j)$.
- $a_{ij}$ - is the number of admitted requests of type $(i, j)$.
- $m_{ij}$ - indicates the number of missed (rejected) requests of type $(i, j)$.
- $U_{avail}$ - is the estimated available utilisation.
- $U$ - is measured (actual) utilisation, which includes admitted user tasks in addition to background tasks assigned by the operating system and the direct tasks not under control of the LCU.

The overall objective of the LCU is to achieve the stabilisation of the CPU utilisation around a set point $U_s$ while managing
individual user/task preferences according to $p_{ij}$. A second controller (UP-CTRL) sets $p_{ij}$, which represent the proportion of the available load that should be allocated to a certain request type $(i, j)$, $\sum p_{ij} = 100\%$ of $U_{avail}$. This policy can be chosen based on task/user importance or other technical/economical data. The LCU is split into two different parts, a deterministic part and a feedback part. The first element (the LC-Logic) is responsible for accepting tasks based on a differentiation policy. The differentiation element however, cannot deal with inaccurate $c_{ij}$, or with a variable load from tasks outside our control ($U_d$). Thus we use a second element to adapt our current CPU claim by using feedback about actually measured utilisation (compared to the desired set point $U_s$). L-CTRL provides adjustments to variations in $U_d$ and $c_{ij}$, and its output is the estimation of the current available load $U_{avail}$. We have tested various P or PI controllers as instantiations of this unit. In LC-Logic the utilisation shares decided by the UP-CTRL are used to compute the available share for each type: $u_{ij} = p_{ij} \times U_{avail}$.

In reality, the mix of incoming tasks may not correspond to the set points $u_{ij}$. If the requests of a certain type $r_{ij}$ are demanding more utilisation from the processor than $u_{ij}$ (i.e. $r_{ij} \times c_{ij} > u_{ij}$), then we have a request excess for the type $ij$. Let us assume we allocate utilisation quotas of size $u_{ij}$ to the incoming tasks. We try to accommodate as many tasks as possible within the allocated quota. Then, the following cases may arise:

1. We have a quota excess for all the types: $r_{ij} \times c_{ij} > u_{ij}$. Then we accept for every $ij$ as much as its quota can accommodate: $a_{ij} = u_{ij} / c_{ij}$.

2. We have a quota shortfall for all types: $r_{ij} \times c_{ij} < u_{ij}$. Then we can accept all incoming tasks: $a_{ij} = r_{ij}$.

3. We have a mix of excesses and shortfalls for different types $ij$. In this case, for each type we accommodate the requests that fit in the allocated quota. If there is free space in some of the quotas, it will comprise an available pool. This pool will then be allocated to the excess requests.
By reallocating the pool, our admission control algorithm is preventing situations in which the requests of some of the types are rejected, while other types cannot fill their quota and the processor becomes underloaded. Further we present the pseudocode for the admission control algorithm. In this algorithm, the pool is allocated proportionally to the requested excess. In the end, if the pool is not big enough some of the tasks will be rejected. Figure 7.3 presents the pseudocode of the pool algorithm.

\[
\begin{align*}
Pool & := 0 \\
Over & := 0 \\
\text{for } i, j := 1 \text{ to } N \text{ do } /\!/\text{loop1: allocating according to quota}\; & \\
& \text{if } r_{ij} \cdot c_{ij} \leq u_{ij} \text{ then} \\
& \quad a_{ij} := r_{ij} \\
& \quad Pool := Pool + (u_{ij} - r_{ij} \cdot c_{ij}) \\
& \quad O_{ij} := 0 \\
& \quad m_{ij} := 0 \\
& \text{if } r_{ij} \cdot c_{ij} > u_{ij} \text{ then} \\
& \quad a_{ij} := u_{ij} / c_{ij} \\
& \quad O_{ij} := r_{ij} \cdot c_{ij} - u_{ij} \\
& \quad Over := Over + O_{ij} \\
\text{for } i, j := 1 \text{ to } N \text{ do } /\!/\text{loop2: reallocating the pool}\; & \\
& \text{if } O_{ij} > 0 \text{ then} \\
& \quad a_{ij} := a_{ij} + (Pool \cdot O_{ij} / Over) / c_{ij} \\
& \quad m_{ij} := r_{ij} - a_{ij}
\end{align*}
\]

Figure 7.3: The Pool algorithm

Although the pool allocation does not correspond to the initial weights between tasks \((p_{ij})'s\), it is a computationally efficient way to allocate all available CPU utilisation. We could allocate the pool according to the initial weights, by iteratively running the first part of the algorithm (loop1). At each additional iteration, \(U_{avail}\) is initialised with the value of the remaining pool from the previous iteration. While improving the differentiation accuracy, the iterative process increases the computational overhead.
Thus, all of our experimental runs use the simple version of the algorithm.

### 7.5 Evaluation setup

In order to evaluate our scheme, different user scenarios have been created and tested. The tests were performed at Ericsson radio Systems in cooperation with Lingvall [56], who implemented the simulator and the traffic generation functions. The simulation environment enables us to test-run the scenarios on an existing simulated model of the target machine together with the relevant operating system and middleware.

The traffic models generated are derived from available data in the telecommunication industry, providing different RNC traffic functions for different types of user applications like voice, mail, SMS and web browsing [56]. As an example we show how we generate a ”web browsing” session. Figure 7.4 shows the states in which a connection can be and the possible transitions, which result in different traffic functions.

![Diagram of a web browsing session]

**WWW Browsing parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intensity (sessions/second/CPU)</td>
<td>Poisson</td>
</tr>
<tr>
<td>Time in DCH per WWW request</td>
<td>Negative exponential</td>
</tr>
<tr>
<td>Number of WWW requests</td>
<td>Geometric</td>
</tr>
<tr>
<td>Delay between WWW requests</td>
<td>Negative exponential</td>
</tr>
</tbody>
</table>

**Figure 7.4:** The “www-browsing” session

Our tests concentrate on overload scenarios. The offered load quickly rises to about 30% over the maximum capacity of the processor. The traffic profiles provide a realistic enough approximation according to our knowledge about 3G traffic behaviour.
7.5. EVALUATION SETUP

We compare our architecture against an existing algorithm at Ericsson, which uses the leaky bucket mechanism. On a sample-by-sample basis, the core difference between the existing and the new algorithm is in terms of the LCU design. In the following experiments we therefore exclude the effect of a long-term adjustment of the system to user policy revisions (UP-CTRL).

![Figure 7.5: Leaky-bucket admission](image)

The leaky bucket type algorithm, used to control the load on the processor, works in the following manner: Whenever a task arrives, first it checks whether enough space is available in the bucket. For each task type a different rejection level can be specified. If the "water" in the bucket is above the specified level the task will be rejected. If it is below, the task is accepted and certain amount of "water" is poured into the bucket. In Figure 7.5, tasks corresponding to level 2 will always be preferred to tasks corresponding to level 1. At regular intervals, "water" is taken from the bucket to emulate a constant outflow. To provide adaptability, the following feedback mechanism is used: the amount of water put into the bucket, when a task is accepted, depends on an average load measured on the processor: \( U_{avg}(k) = w1 \times U(k) + w2 \times U_{avg}(k - 1) \), where \( k \) represents the number of the control interval. If \( U_{avg} > U_s \) then the amount is decreased with a certain step, if \( U_{avg} > U_s \) it is increased with the same step.
7.6 Simulation results

Recall that our load control system has two goals. First, to avoid reaching CPU load levels beyond a particular threshold, and second, to differentiate between the user/task types as specified by a given policy. Figure 7.6 shows how our architecture compares to the leaky bucket architecture in the context of a realistic traffic scenario. The load which is allowed on the CPU depends only on the feedback mechanisms of the architectures, thus the results show how successful they are in regulating the accepted load. The Y-axis shows the accumulated overload beyond the given threshold ($U_s$), expressed in terms of percentages. This graph shows that the leaky-bucket architecture overloads the processor more often than our controller.

![Graph showing cumulative overload beyond the desired threshold](image)

Figure 7.6: Cumulative overload beyond the desired threshold

In order to better explain Figure 7.6, Figure 7.7 shows the load on the processor generated by the leaky bucket architecture compared with our architecture (pool + P-controller). The amount over the 80% line is the overload presented in the previous figure. Figure 7.7 also shows that the P-controller is quicker to adapt to
7.6. SIMULATION RESULTS

Figure 7.7: Load comparison
the changes in $U_d$, which increases the overload protection of the processor. On the other hand, the leaky bucket architecture has slightly less rejections than ours. We argue that this is because it also allows more load on the processor. Our experiments show that no matter which differentiation policy is used, the use of feedback control ensures a more stable behaviour in the light of inaccuracies in the $c_i$ and $U_d$ estimates.

Let’s now look at the differentiating behaviour of the proposed algorithm. The relative prioritisation of the allocation quotas is illustrated for the 50/50 case in Figure 7.8, where the number of accepted and rejected tasks is plotted for one user type. With a 50/50 we expect half of the CPU share to go to new connections and half to channel switches. In this case the new connections are not numerous enough to occupy the entire allocated share, thus the pool is reallocated to the channel switches. This results in the number of accepted channel switches to be above the 50% share. We would expect that the number of rejected new connections to be zero, but there are few such rejections due to estimation errors of the adaptive part.

We have also ran experiments to separate the effect adaptive part, that deals with uncertainties in $c_{ij}$ and $U_d$, and the deterministic part. These experiments show that the $U_d$, being at times very oscillative, has a stronger impact on the results than the $c_{ij}$. However, the experiments also show that the adaptation element can be tuned with the right choice of coefficients.

7.7 Summary

This chapter presented adaptive mechanisms for load control in an RNC node of a 3G network. The main goal was to provide a protection scheme for the processor while rejecting as few control tasks as possible, and ensuring that in case of overload, different task types are accepted based on a policy decided by the service provider. This should prevent that a user, who was accepted into the system based on certain QoS parameters and bandwidth availability, is later rejected in a load control algorithm. System
Figure 7.8: Behaviour with a 50/50 differentiation policy
behaviour under traffic overloads should be consistent with the provider policy. The presented architecture can be partitioned in two main components. The adaptive part is a feedback controller, whose function is to deal with the uncertainties in our model, e.g. the exact execution time of the controlled tasks and the load on the processor generated by tasks outside our control. The deterministic part is a share-based differentiating algorithm, which ensures that the available load is partitioned corresponding to a certain policy. The presented architecture successfully protects the processor from overload, and also enforces the specified QoS.

We have compared our proposal with an existing regime that is based on the leaky bucket concept. While this algorithm is satisfactory in the current industry setting, it enforces absolute priorities among task types. If the need for relative prioritisation arises in future business models, the absolute priority may not suffice. In our case, we formulate QoS requirements as relative priorities between different task/user types, and enforce them by quotas for the different types. our implemented simulator and traffic generator. While the simple pool algorithm has a low complexity, it will not guarantee strict priorities (shares) among tasks, due to the pool reallocation mechanism. The latter can be improved though, by running several iterations of the algorithm.

The next issue was the adaptive character of the architectures. Confronted with a strong oscillating $U_d$, neither the P-controller nor the leaky bucket estimator show an ideal behaviour. The P-controller has the drawback that it follows the oscillative character of the $U_d$, one control interval behind. On the other hand the averaging estimator used in the leaky bucket architecture is at times too slow to react, leading to the possibility of overload of the processor. To improve the behaviour of our control scheme it would be necessary to have a better estimation of $U_d$. This entails treating traffic functions and management tasks differently (e.g. delaying management tasks selectively). In particular, there are tasks whose rejection increases the short-term load (rejecting channel switches means that the connection is dropped, and clean-up has to be performed). The load generated by the rejection of
these tasks could be taken account of in the estimation of future $U_d$. 
Chapter 8

Conclusions and Future Work

8.1 Conclusions

It is still debatable whether QoS mechanisms will play a significant role in the Internet, due to cheap overprovisioning or preference for flat rates [98]. Resource constraints in wireless networks on the other hand seem unavoidable, and the market is likely to allow for price discrimination [99].

The effort in this thesis was directed into developing a comprehensive bandwidth allocation scheme for wireless networks. Besides being subjected to resource constraints, wireless networks represent a class of distributed systems with a higher degree of unpredictability and dynamic change as compared to their wireline counterparts. New types of adaptive, QoS-aware resource allocation schemes are needed to improve the behaviour during periods of overload. QoS differentiation is needed in providing support for different types of applications and services, with different importance and resource needs. Even in non-commercial cases such as military use or disaster management, active differentiation respecting the criticality of the different communications is needed for the success of the mission.
We propose several bandwidth allocation schemes, presented in Chapters 4, 5, and 6. The first scheme is employed in the context of a cellular wireless network. Then we extend the study to a hybrid wireless setting, and finally to a pure ad hoc environment. We aim for a global optimisation policy, abstracting away lower-level enforcement mechanisms. The goal of the policy is to maximise the total utility of the system, expressed as the sum of all connections in the system. The connection utility for different bandwidth allocation levels is specified using a utility function.

**Resource reservation versus resource adaptation**

In an open, dynamic system a resource allocation decision might be optimal at a certain time point, but as new requests arrive it might become quickly suboptimal. Now the question is: should we keep the suboptimal allocation for the old connections and keep the same QoS for the whole duration of the connection, or should we reallocate? If we reallocate, we would break the ongoing QoS contract. Moreover, due to dwindling resources, a reallocation might be unavoidable. The novelty of our approach is that we combine both the previous choices in a consistent manner. We synthesise the consequences of reallocation for different types of applications, and use this information while performing a periodic allocation/reallocation optimisation.

Basically the scheme presented in Chapter 4 allows the consistent treatment of applications that have rigid reallocation requirements (e.g. real-time) in a system where periodic optimisation is performed. Please note that the reallocation sensitivity is decoupled from the importance of the application. Admission is based solely on importance, but subsequent reallocation decisions are treated suitably.

Due to the characteristics of mobile networks, bandwidth availability cannot be guaranteed to connections. Thus the aim of the scheme is not to enforce the provisioning of a certain QoS level, but to provide a flexible allocation strategy where constant reallocation strives to maximise system-wide utility. Nevertheless, our scheme is able to ensure relative guarantees, which means that
every user is ranked according to the generated utility, and, as long as resource is available, it will be allocated according to the user’s requirements.

Finally, we consider the overhead generated by the reallocations. For example CPU time (adapting the application to the new mode) and signalling traffic will increase when executing the reallocation decisions. We choose the number of bandwidth reallocation as a metric for characterising such demands, and show that we can greatly reduce the strain on the system with only a small decrease in the generated utility. Consequently, this new algorithm could be employed to save much needed resources.

Modelling non-critical resources as costs

The second allocation scheme is set in a hybrid wireless environment, i.e. a cellular core network extended with ad hoc paths. The bandwidth of the cellular network is the bottleneck resource compared to the bandwidth of the ad hoc extension, and this simplifies the allocation problem. While non-critical, the other resources are of concern especially since in the multi hop path, other people’s equipment is used to relay the connection. Thus we model the usage of these resources as a cost function (e.g. proportional to the incentives to be paid to the users for relaying) and include it in the optimisation problem.

While the allocation at each base station is done similarly to the cellular scheme, a mobile station has access and must choose between several possible base stations. A distributed algorithm is used for solving the latter problem. The experiments show that the distributed algorithm has at worse 12% lower performance than the optimal one, while being several orders of magnitude faster. Compared to a pure cellular scenario, when dealing with uneven traffic, the scheme provides load balancing among base-stations and fault tolerance in the case of coverage failure.
Distributed price-based allocation for ad hoc networks

In the third case, we study bandwidth allocation in a pure ad hoc environment. We allocate resources in a multi hop environment, where the bandwidth of an end-to-end connection is constrained by its most restricted link. While in the wireline setting each link is considered as an independent resource, in wireless networks links that are near to each other are contending for the same channel bandwidth. Thus, a bandwidth resource represents a maximal clique of mutually contending links.

We first show that, assuming that routes are fixed, the utility-based bandwidth allocation problem in a multi-hop wireless network can be formulated as a linear programming maximisation problem. Although centralised and infeasible for online allocation in an ad hoc environment, the LP solution provides an upper baseline for comparisons.

We then propose a distributed allocation algorithm that bids for resources on the multi hop path on behalf of every flow. The bid is constructed depending on the “shadow prices” of resources, and the “utility efficiency” of the flow. We borrow the concept of shadow price from the dual of the allocation problem, where it is used to evaluate resources.

As mentioned previously our allocation scheme uses periodic reallocations to optimise performance. Thus, we use the iterative process to both adapt to changes in the network and also to recalculate and improve our estimation of shadow prices. More specific, the bids on behalf of a flow are constructed using the shadow prices from the previous allocation, and the shadow prices of a resource are derived from the current bids.

Experimental results show a very good behaviour of our distributed allocation algorithm, that is within 10% of the behaviour of the optimal centralised LP solution. In the literature there is only one other type of distributed price-based allocation algorithm described [47, 48, 90], where the allocation of a flow is constantly changed depending on the congestion (i.e. price) of the resources. More studies are needed to compare the performance of the two algorithms for flexible applications. Regarding more rigid applic-
8.2. FUTURE WORK

Ations, where reallocation can be disruptive, our algorithm poses a real advantage, since a) the allocation is fixed for at least one period, and b) the reallocation decision takes account of the reallocation effects on the utility.

Dealing with uncertainties

Resource allocation in wireless networks is subject to several uncertainties, ranging from unknown mobility, changing traffic patterns, to inaccurate knowledge of resource requirements. Thus in addition to a global allocation policy, we have studied the advantages of using feedback to control the resource allocation process.

The problem studied is the overload protection of a RNC CPU, on which the availability of the surrounding network depends. The traditional load control method that rejects only new connections is not sufficient for an UMTS environment, in which already accepted connections may generate enough control tasks to overload the processor. Task type differentiation (e.g. channel switch function are more important than connection setup functions) and user type differentiation (e.g. emergency calls, or golden vs. normal user) is employed, and provide the means to enforce the network operators QoS policy.

The feedback-control used in our scheme successfully protects the CPU from overload and performs better than an existing method suggested by the industrial partner. However, better modelling of the CPU load, which means a characterisation of the tasks outside our control, and the use of more advanced control techniques, could further improve the performance of the controller.

8.2 Future work

A next step in continuing the work presented in this thesis, is to study the convergence properties of the multi hop allocation algorithm together with the benefits and overheads of using several iterations at each allocation point in order to further improve the allocation decision. The behaviour of the routing part of the
The scheme we have presented in the ad hoc part needs synchronisation among the nodes in the network to provide a consistent and fixed allocation for the end-to-end connection. The biggest problem in this case is not clock synchronisation, but the delays induced by the signalling overhead, that in the end could lead to allocation inconsistencies along the multi hop path. Moreover, in the actual setting, new or rerouted connections have to wait until next allocation period to have any resources allocated. To alleviate this we propose to use feedback control for controlling the amount of a bandwidth reservation pool. Then, new or rerouted connections can be accommodated immediately. Moreover, the allocation at the clique-resources can be performed asynchronously. Obviously, only connections whose bid is higher than the current allocation threshold should be accepted in this way.

Finally, the challenges of implementing the ad hoc allocation scheme on real networks must be addressed. Ideally, by correctly communicating the allocation values with the source of the connection, QoS should be preserved regardless of the lower layer QoS enforcement mechanisms. In practice though, the enforcement choices cannot be neglected, moreover taking into consideration that in open networks some nodes might not want to cooperate, trying to get as many resources as possible or behave downright maliciously.
Bibliography


[59] 3GPP, “Quality of service, concept and architecture,” TS 23.107, June 2002.

Western Simulation Multiconference - Communication Networks and Distributed System Modeling and Simulation, June 2001.


Linköping Studies in Information Science


