Mälardalen University Licentiate Thesis
No.84

Resource Management Framework for Distributed, Heterogeneous Systems

Larisa Rizvanovic

April 2008

School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden

ISSN 1651-9256
ISBN 978-91-85485-78-9
Printed by Arkitektkopia, Västerås, Sweden
Distribution: Mälardalen University Press

Abstract

In distributed heterogeneous computing environments, such as in-home entertainment networks and mobile computing systems, independently developed applications share common resources, e.g., CPU or network bandwidth. The resource demands coming from different applications are usually highly fluctuating over time. For example, video processing results in both stochastic fluctuations, caused by different coding techniques for video frames, and structural fluctuations, due to scene changes. Similarly, wireless networks applications are exposed to long-term bandwidth variations caused by other application in the system that are using the same wireless network simultaneously, and short-term oscillations due to radio frequency interference, like microwave ovens or cordless phones. Still, applications in such open, dynamic and heterogeneous environments are expected to maintain required performance levels.

In this thesis, we look into solutions for efficient transport of video streams with acceptable playout quality in home networks, which requires management of both networks and CPUs. We propose a framework for efficient resource management for streaming in heterogeneous system, called the Matrix. The Matrix is based on a global abstraction of device states, which reduces system state information and decreases overheads for its determination and dissemination. It provides access to the entire system state in acceptable fresh way, enabling system wide optimized decisions to be taken.

Moreover, we use the Matrix framework as the platform to develop a method for an efficient Quality-of-Service (QoS) provision and adaptation in dynamic, heterogeneous systems. QoS adaptation is one of the crucial operations to maximize overall system quality as perceived by the user while still satisfying individual application demands. It integrates local QoS mechanisms of the involved devices that deal mostly with short-term resource fluctuations, with a global adaptation mechanism that handles structural and long-term load variations on the system

level.

We have illustrated the effectiveness of our QoS adaptation approach in the context of video streaming. However, we do not see any limitation to expand the usage of our approach to the health sector, or some other community social/industrial applications. Resource management and QoS adaptation are required whenever we are surrounded with heterogeneous, mobile, and dynamic environment.

Swedish Summary - Svensk Sammanfattning

I distribuerade heterogena miljöer, som hemmanätverk och mobila datorsystem, oberoende utvecklade tillämpningsprogram måste dela gemensamma resurser, till exempel, CPU, eller nätverksbandbredd. De olika tillämpningsprogrammen har ofta högt varierade krav på resurser. Exempelvis, bearbetning av en video kan resultera i både temporära variationer, orsakade av olika kodnings tekniker för videoramar, samt strukturella variationer, på grund av scenändringar. På samma sätt, kan trådlösanätverk utsättas för långtids- eller korttidsvariationer på grund av olika störningar. Ändå, tillämpningsprogram i sådana dynamiska och heterogena miljöer förväntas att behålla det garanterade prestanda.

I denna avhandling har vi studerat lösningar för effektiv transport av videoströmmar med god upspelningskvalitet i en heterogen miljö, vilket kräver hantering av både nätverk och processorresurser. Vi presenterar ett ramverk för tillämpning av realtidsresurshanteringsmetoder för direktuppspelning av video in en heterogen, mobil miljö. Ramverket baseras på en global abstraktion av enheters tillstånd, vilket i sin tur reducerar omfattningen på systems statusinformation, samt minskar kostnader för dess bearbetning. Ramverket möjliggör integration av olika enheter från olika tillverkare med olika krav och kapaciteter.

Förutom hantering av resurser, kvalitetsmedveten anpassning är en av de avgörande funktionerna för att maximera användarens kvalitetsupplevelse, och samtidigt uppfylla de specifika krav som varje tillämpningsprogram har. I denna avhandling har vi utvecklad en metod för en effektiv kvalitets tillhandahållande och anpassning av tillgängliga resurser i dynamiska heterogena system. I metoden hanteras temporära och korttidsvariationer lokalt på enskilda enheter, medan strukturella och långtidsvariationer är objekt av en global kvalitetsanpassning, där alla enheter i systemet är involverade.

Vi har valt att utvärdera våra metoder i multimediasammanhang, men vi ser inga hinder att tillämpa våra lösningar inom vården, eller andra samhällens sociala/industriella användningsområde. Hantering av resurser och kvalitetsmedveten anpassning behövs närhelst vi är om-

givna med heterogena mobila miljöer.

To my beloved sons, Teo and Tim!

Preface

Ignorance is bliss. Little did a 7 year old girl from Mostar (Bosnia-Herzegovina) know that next 28 years of her life are going to be spent in a school bench, on the sunny September day in 1978, her first day in school. 28 years, how come?! Well, the reason for this quite long education was not bad grades, as one could guess, but the turmoil of war that hit the Balkans in the nineties. But, all things considered, it was not so bad after all. Not so many people have attended high school twice, or have studied both bridge and computer design, as I did.

Hence, I have really many people to thank for their support during my many years of education. But I will focus on just my time as graduate student here at Department of Computer Science and Electronics, Mälardalen University.

The work presented in this thesis would not have been possible without the help of my supervisors. I want to thank my main supervisor Gerhard Fohler, for accepting me as one of his students, and for his believe that I can manage this journey. A big thank you goes to Damir Isovic, my second supervisor, who has been a great support during this last year. And of course, many thanks to the rest of "Salsart" group, Radu Dobrin, Tomas Lennvall, and Robert Bäckgren for many interesting discussions, your support, and good laughs.

Then, many hugs go to Monica Wasell, Harriet Ekwall, Else-Maj Silén, Ewa Hansen, and Jonas Neander for so many wonderful coffee break moments. Of course, I would like to mention the rest of my colleagues here at the department, especially Dag Nyström and Thomas Nolte, who I met back in 1997 when we started our undergraduate education here. Thank you all for making this department fun to work in.

Furthermore, I thank you to my friends, Seija Tasala and Mariana Olsson, for their encouragement to continue with this work, even when things did not work so well.

Finally, and most important, I thank my great family. My all love goes to my beloved boys, my sons, Teo and Tim, and my husband, Ned. A big thank you to my wonderful mom, dad, and my big sister, for being always there for me and my boys. Thanks my parents-in-law, for believing in me.

This work has been supported by Mälaradalen University's grant for a female graduate student.

Larisa Rizvanovic
Västerås, March 2008.

List of Publications

1. *Integrated Global and Local Quality-of-Service Adaptation in Distributed, Heterogeneous Systems*, Larisa Rizvanovic, Damir Isovic, Gerhard Fohler, International IFIP Conference on Embedded and Ubiquitous Computing (EUC-07), LNCS Lecture Note, Taipei, Taiwan, December, 2007.

2. *The MATRIX - A Framework for Real-time Resource Management for Video Streaming in Networks of Heterogenous Devices*, Larisa Rizvanovic and Gerhard Fohler, The International Conference on Consumer Electronics 2007, Las Vegas, USA, January 2007.

3. *Real-time Architecture for Networked Multimedia Streaming systems*, Larisa Rizvanovic, Gerhard Fohler, MiNEMA Workshop, 2006, Leuven, Belgium, February, 2006.

4. *The MATRIX: A QoS Framework for Streaming in Heterogeneous Systems*, Larisa Rizvanovic and Gerhard Fohler, International Workshop on Real-Time for Multimedia, in conjunction with ECRTS04, Catania, Italy 2004.

5. *Integrated Quality-of-Service Adaptation in Distributed, Heterogeneous Systems*, Larisa Rizvanovic, Damir Isovic, Gerhard Fohler, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-224/2008-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, 2008.

6. *The Matrix - A framework for real-time resource management for video streaming in networks of heterogenous devices*, Larisa Rizvanovic, Gerhard Fohler, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-216/2007-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, October, 2007.

7. Submitted, not included in this thesis: *Double-Stimulus Subjective Quality-of-Service Assessment for Video Applications*, Riasat Abbas, Larisa Rizvanovic, and Damir Isovic, IEEE International Conference on Multimedia & Expo, June 23-26, Hannover, Germany.

Contents

List of Figures

Chapter 1

Introduction

Nowadays, most home entertainment devices, such as TV sets and VCRs, are fully digital, demanding computing methods to match strict temporal demands of audio and visual perception. Consequently, the concept of "one cable - one box" is being replaced with pictures and videos available where and when demanded in-home. Similarly, video transmission and communication over mobile phone is already starting to become commonplace. Thus, our homes will be probably equipped with in-home networks where wireless connected devices will be overrepresented in order to provide a responsive and supportive environment, known as Ambient Intelligence (AmI). Ambient Intelligence is when an existence of digital environment is sensitive, adaptive, and responsive to the presence of people [38].

The work presented in this thesis has started as part of the FABRIC EU IST project [1]. The aim of the project was to develop an architecture in which several interoperability standards and technologies in the home networking context can be integrated. In addition, FABRIC aimed to handle the complete network to satisfy End-to-End Quality of service (QoS) requirements.

[1] FABRIC: Federate Applications Based on Real-time Interacting Components (EU IST project IST-2001-37167)

1.1 Motivation

This work deals with real-time architectures for networked multimedia streaming systems. Key challenges to be addressed include specification of stream and resource characteristics, high demands on processing and timely delivery of multimedia streams, wireless communication between devices and transmission of streams, and architectures for the integration of numbers of devices from various manufactures with diverse demands and capabilities. Projects in this area involve work on developing new algorithms which can match the varying multimedia streams with the varying network and CPU resources, as well as implementation work, developing and implementing new architectures for system capable of handling such networked streaming issues.

Figure 1.1: Home networks environment

In distributed heterogeneous environments, e.g., home networks (see

Figure1.1), we have:

- Heterogeneous system; Presence of different local schedulers for CPUs and networks on diverse devices.

- Limited resources; Monitoring and management orders have to be transported over the same resources as streams, potentially incurring high delays, and bandwidth cost at streams' expense.

- Highly fluctuating resources; Processing of a media application is highly dynamic due to the dynamic nature of the audio/video media content. Thus, when processing a video stream, there are two types of load fluctuation due to data dependency: stochastic and structural [36] (see Figure 1.2). A stochastic load, can be caused by different coding techniques for video frames. A structural load often occurs due to a scene change. Similarly, wireless networks applications are exposed to long-term bandwidth variations caused by other applications in the system that are using the same wireless network simultaneously, and short-term oscillations due to radio frequency interference, like microwave ovens or cordless phones.

Still, applications in such open, dynamic and heterogeneous environments are expected to maintain required performance levels. Efficient transport of streams with acceptable playout quality requires management of both networks and CPUs. As resources are typically limited in home environments, guarantee mechanisms for continuous stream transport are demanded.

One of the key issues for resource management is an efficient representation of the fluctuating system state and resource allocation decisions, to provide a small interface to decouple device scheduling and system resource allocation. In addition, Quality-of-Service adaptation is an important operation to maximize overall system quality as perceived by the user, while still satisfying individual application demands.

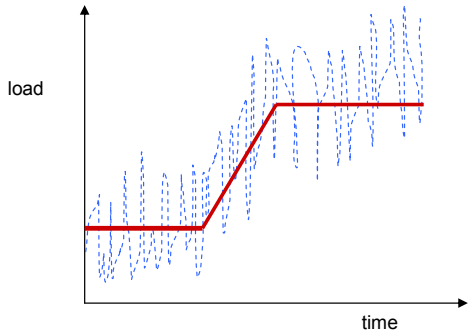

Figure 1.2: Two types of load fluctuation; stochastic (scribble line) and structural (straight line)

1.1.1 Research method

As we mentioned, our work has started as a part of FABRIC project, where our industrial partners, like Philips and Thomson, had a tangible influence on the project's problem description. We have started with one consumer electronics market's well defined problem and made many cycles from theories down to observation and back up again to theories. Thus from the very beginning, we use combination of the deductive and inductive methods.

Simplified, our approach can be divided in three parts:

1. Theoretical level; Here we will work on developing new algorithms.

2. Experimental aspects; Many parameters and trade-offs for the algorithms have to come from experiments and cannot be calculated

3. Implementation work; In order to evaluate the hypothesis and hopefully give answers to the research questions

1.2 Related work

1.2.1 Quality of Service

Quality of Service (QoS) is a wide term, often used to describe overall experience an application or a user will receive over a network. The literature does not offer one common definition of QoS. In [5] QoS is defined as:

"The set of those quantitative and qualitative characteristics of a distributed multimedia system, which are necessary in order to achieve the required functionality of an application."

According the recommendation of the Telecommunication standardization sector of International Telecommunication Union (ITU-T)[23], Quality of Service is defined as:

"The collective effect of service performance which determines the degree of satisfaction of a user of that service".

Thus, for different levels of the multimedia system, different definitions of QoS are given.

When designing multimedia streaming systems, Quality of Service is one of key issues. QoS parameters in such systems have to be presented in all components, from an application to a communication level in order to insure a certain level of QoS to a user. However, at different system layers (e.g., application, operating systems, network) quality of service is formulated by using different parameters. The translation between QoS parameters, sometimes also called QoS mapping, on various levels of a QoS architecture is needed. QoS mapping is also required in order to reserve the appropriate amount of resources on each layer of QoS framework. In [17, 45, 14] different mapping mechanisms that translate representation of QoS at different system layers (i.e., application, network) are presented.

In the following subsections, we are going to give a short introduc-

tion to work done on specifying QoS parameters at different layers, and QoS frameworks.

User perceived QoS

The ultimate measure of QoS in multimedia application is user satisfaction. Hence, user perceived QoS is becoming more important research field for industry. Although, user perceived quality is beyond the scope of our work, we have to be able to define and choose different video qualities. In the literature, methods for quality assessment of video are divided in two categories: subjective and objective. Subjective methods consists of the compilation and statistical analysis of sample ratings generated by humans. On the other hand, objective video quality assessment uses quality metrics that can predict perceived video quality automatically.

Subjective video assessment The International Telecommunication Union (ITU) has proposed Mean Opinion Score (MOS) as a measure for a subjective quality assessment of video [24]. The MOS is obtained from a number of human observers who rank the perceived quality of the shown media content from “worst‘ to “best“. The MOS has been regarded for many years as the most reliable form of quality measurement.

Nevertheless we can find many researchers questioning the ITU recommended methods for subjective quality assessment of video. In [44], authors argued that the ITU methods are not suitable for assessing the quality of multimedia application, and show that the subjective nature of the MOS makes it not very robust. They also strongly advise not to use these methods to assess subjective quality required by multimedia applications developed today, because they can be misleading in some cases. According to the same paper, speech and video are multidimensional phenomena. Thus, there are multiple factors that can influence users’ perception of video and speech. Measurement of perceived media quality has to be done as function of physiological responses. In [7],

the authors state that instead of trying to generalize users' QoS requirements, it is better to use different Human Computer Interaction (HCI) methods in order to answer certain questions about such requirements. Different HCI methods can be used to answer specific questions concerning QoS requirements in different context. In [7] a stress has been also put on the fact that there are situations when people can not report a difference between two frame rates, but the difference has been registered physiologically. In [29] psycho visual experiments designed to evaluate the perceived quality of low bit rate video are described. The subjective evaluation was performed using the Single Stimulus Continuous Quality Evaluation SSCQE [22]. SSCQE was designed to perform time efficient subjective quality evaluations of digital services, in conditions near to ones found in the home environment. In [22], an analysis of viewer responses to different artifacts across the range of possible coding conditions and content is presented. In general, the authors found that the optimal frame rate, given a constant bit rate, appears to be a function of the type of motion in a sequence. Thus, sequence with jerky motion benefited from the increased spatial quality at lower frame rates, while the perceived quality of sequences with smoother motion where in general unaffected by changes in frame rate.

Video streams have different bit rates. A relation between a stream bit rate and the required bandwidth to carry the stream is obvious: the higher bit rate, the higher bandwidth is needed. However, a relationship between video quality and the corresponding bit rate is not clear. [35] analyses a relationship between the user-perceived quality and the average encoding bit rate for variable bit rate MPEG-2. The major conclusion of the paper is that the image quality can not be improved by increasing encoding bit rate only. Increasing the bit rate above a certain threshold can result in video degradation. For a given packet loss ratio, a quality-optimal coding rate has to be found.

Objective video quality assessment Objective video quality assessment research aims to design quality metrics that can predict perceived video quality automatically. A mathematically estimation of the im-

pairment introduced to video during compression is often used method within this quality assessment.
Peak-Signal-to-Noise-Ratio (PSNR) or Signal-to-Noise-Ratio (SNR) is one video quality metrics often used for audiovisual signals. PSNR analysis uses a standard mathematical model to measure an objective difference between two images. It is commonly used in the development and analysis of compression algorithms, and for comparing visual quality between different compression systems. However, many researches have shown that PSNR is uncorrelated with the human visual system and cannot be trusted [35, 43], as it does not take a visual masking into consideration. In other words, every damaged pixel contributes to a decreased PSNR, although this error can not be perceived. In [43], a general quality metric for video, termed Moving Pictures Quality Metric (MPQM) is proposed. MPQM provides a metric between 5 (excellent) to 1(bad) to express the quality of image streams. The model is based on the properties of human vision. It considers visual masking technique to effectively take into account error concealment techniques in the QoS result and directly targets video rather than single image quality. MPQM method provides a formula to estimate how packet loss impacts the quality of image streams.

Application QoS

Application QoS parameters capture an application's QoS requirements. Depending on a type of application, application QoS requirements can vary, e.g. QoS requirements for an e-mail application are not same as for a streaming application. These two types of application have quite different demands on bandwidth, as well as their sensitivity to delay, jitter, and data loss are not the same. In [30] as application-level performance metrics are identified: throughput, latency, availability, data loss, jitter, and security. In [31] some additional application parameters like colour, bit length, sampling rate are mentioned.

System and network QoS

From a user point of view, end-to-end application QoS parameters are the most important ones. For the user, system QoS performance is hidden. However, system parameters depicts communication and operating system requirements, needed by application. Thus, at system layer, QoS parameters can be buffer size, process priority, scheduling policy, and time quantum.

Examples of network QoS parameters are bandwidth, throughput, loss, delay and jitter [33]. They can be expressed in terms of number of packets, periodicity of packets, and burst size [13].

1.2.2 QoS Architectures

A fair amount of work has been done on satisfying quality of service requirements on different architectural levels, e.g. network or operating systems. However, providing QoS guarantees in distributed multimedia systems has to be done on an application-to-application basis, and a QoS awareness should be present at all parts of a stream management, i.e., from a media source to a playout devices [3]. In the literature several architectures have been proposed to realize end-to-end QoS for multimedia applications. In the following, we briefly describe some of these.

The OMEGA Architecture is designed and developed at the University of Pennsylvania [32]. It is an end-point architecture designed to provide real-time guarantees in networked multimedia systems. The authors have identified application QoS requirements and investigated how to satisfy those with help of global and local management. In OMEGA, application requirements, expressed as QoS parameters, are negotiated, and if possible, guarantees are made at several logical levels, e.g. between application and network. The OMEGA has been designed on assumptions that a network subsystem provides bounds on delay, errors, and can meet bandwidth requirements, likewise an operating system can provide run time QoS guarantees.

QualMan is a QoS-aware resource management platform that was designed based on the knowledge and experience gained during design-

ing the OMEGA architecture [11]. According to [11], research around OMEGA architecture was more concentrated on QoS management than on resource management. The authors of QualMan are convinced that QoS management is only part of the end-to-end QoS solution and in order to provide end-to-end QoS guarantees, a QoS-aware resource management is needed. In QualMan, a resource manager allows one application to specify the desired quality in terms of CPU, memory, and communication QoS parameters. The resource model in this architecture has the resource broker that provides negotiation, admission, and reservation capabilities over the shared resources, i.e. CPU, memory, and network.

Another interesting work is End System QoS Framework, developed at Washington University [15]. It provides QoS guarantees within the end-system for networked multimedia applications. This framework consists of QoS specification, QoS mapping, QoS enforcement and protocol implementation. QoS specification, expressed in a few parameters, depicts flow requirements at application level. Then QoS mapping part, translate those parameters into resource requirements. Finally, QoS enforcement is involved by providing real-time processing guarantees for media transfer.

At Lancaster University work has been done on the Quality of Service Architecture (QoS-A), which is a layered architecture of services and mechanisms for quality of service management and control of continuous media flows in multiservice networks [6]. Flow, service contract and flow management are key notation used in QoS-A. A flow comprises production, transmission and rendering of media streams. Service contract is an agreement of QoS levels between users and providers. The contracted QoS levels are monitored and maintained during the flow management phase. QoS-A takes into consideration both a user-level specified QoS and mechanisms for resource reservation at a network level.

MASA (Mobility And Service Adaptation) is a comprehensive end-to-end QoS architecture [25]. The aim of MASA is to support user policy-controlled media transmission and processing in heterogeneous

environments. The MASA framework has following main features: an overall management system for end-to-end QoS, mapping of user QoS policies into appropriate QoS parameters for the underlying layers, and a QoS API that makes developing of QoS-enabled applications easier.

In [10] one additional QoS framework, called QoSMF is presented. In order to handle the heterogeneity in distributed multimedia applications, the authors have implemened QoSMF integrated with CORBA A\V STREAM architecture.

A distributed QoS management architecture and middleware that manages a multidimensional aspects of QoS is presented in [41]. This framework provides end-to-end QoS management services that cover QoS characterization and specification, end-to-end QoS negotiation, and end-to-end establishment. Due to the fact the authors consider different dimensions and measures of QoS and also give some concrete examples of mapping function (or as they call it, a reward function), this work appears to be of quite interest for our future work.

In [36] one QoS-based resource management framework for Ambient Intelligence has been presented. It combines resource reservation and application adaptation in a multi-layer QoS architecture, that addresses terminal and network resources, and takes energy issues into account. This framework has ambition to optimize system utility to provide cost-effective systems that are robust, predictable, and stable, enabling QoS tradeoffs.

A great work has been done on a software environment for ubiquitous computing and Ambient Intelligence within the IST project OZONE. Work description for the OZONE project was closely related to the Fabric project. One of objectives of OZONE project is "to specify and implement a generic architecture/framework that will support the effective use and acceptance of ambient intelligence in the consumer domain" [37]. One of tasks within the OZONE architecture was the QoS management for multimedia application. One of results of the OZONE is platform architecture for the application of Consumer Ambient Intelligence. This platform has adaptive media algorithms, and resource and QoS management layer.

SURPASS Home entertainment solution is a product of Siemens. Their strategy aims at provision of seamless interoperability between all intelligent devices in the home [4]. According to Siemens, SURPASS Home entertainment provides users with an end-to-end tested and guaranteed solution. This architecture consists of application control, digital rights management, video servers, video head-ends (i.e. units that convert a television signal to an IP stream, and also encode the signal in real time according to MPEG-2 or MPEG-4), home gateways, bandwidth resource control, and communication controllers. Unfortunately, Siemens does not give a detailed description of SURPASS, hence the parts of SURPASS architecture that handle resource management and streams specifications, remain unknown for us.

However, most of these comprehensive end-to-end QoS architectures, with the exception of [36, 37, 4], are mostly designed to work over networks like ATM or the Internet with Integrated Services (IntServ) and Differentiated Services (DiffServ) support, i.e. networks that can provide guarantees on bandwidth and delay for data transfer. As we mentioned before, our work is performed in the home environment domain, where we have heterogeneity (different devices, communication media and middleware), and limited resources (especially wireless networks). Therefore, we do not make any assumptions regarding ability of the underlying system (OS or network) to offer QoS guarantees.

While architectures like [25] give an overall management system for end-to-end QoS, covering all aspects from a user QoS policies to network handovers, in our work we focus on QoS management and resource adaptation in application domain.

1.2.3 QoS Adaptation of fluctuating resources

QoS adaptation involves monitoring and adjustment of resources and data flows in order to ensure delivering of certain performance quality level to the application. This can be done *locally* on a device, i.e., local resource adaptation mechanisms on devices detect changes in resource availability and react to them by adjusting local resource consumption on host devices, or *globally*, on the system level, i.e., the QoS adaptation

is performed by a global QoS manager with a full knowledge of the system resources. The first approach has the advantage that the application can use domain specific knowledge to adapt its execution to the available resources. For example, in a video streaming application, this could be achieved by decreasing the stream bit rate or skipping video frames. On the other hand, a global resource management is aware of the demand of other applications and it has an overview of the total resource availability on the system level. In this way, it may reassign budgets, or negotiate new contracts to maximize system overall performance. Comprehensive work on application-aware QoS adaptation has been done in [27, 34]. Both these works make a separation between the adaptations on the system and application levels. While in [27] the application adjustment is actively controlled by a middleware control framework, in [34] this process is left to the application itself, based on upcalls from the underlying system.

Classical control theories have been also examined for QoS adaptation. Thus, in [18] the authors use a feedback mechanism based to control the bandwidth of real-time multimedia applications according to network load. The presented algorithm is designed for RTP protocol, and it uses available information in so called receiver reports (e.g. packet loss, and delay jitter) to estimate network state and make bandwidth adjustment. A work presented in [28] shows how an application can be controlled by a task control model. Method presented in [42] uses control theory to continuously adapt system behaviour to varying resources. However, a continuously adaptation maximizes the global quality of the system but it also causes large complexity of the optimization problem. Instead, we propose adaptive QoS provision based on a finite number of quality levels.

The challenges of providing QoS for video streaming over wireless network are discussed in [26]. Furthermore, an architecture is proposed to deal with the unreliability of the wireless network, by adapting the video streams' transmission rate to the varying bandwidth. This work has been of a high interest to us, and we have adopted the parts of this architecture, such as the bandwidth prediction and traffic shaper, in our

work (see Chapter 4 and Appendix A).

1.3 Contributions

As mentioned, to execute applications in open, dynamic, heterogenous environments, like home networks, while still guaranteeing the required qualities, is a challenging task. Many technical and research questions have to be solved in order to give a complete end-to-end solution. Although our ambitions, from the beginning, was to provide a design for a comprehensive adaptive framework for streaming, and to provide answers to all questions that we ran into, we have realized that we need to limit ourself to a few research questions. Hence, in our work we have focused on resource management and QoS adaptation problems, while many other questions (like user perceived quality) are left outside of the scope of this thesis. Consequently, in order to pursue with our work we have had to make a few assumptions.

1.3.1 Assumptions in the thesis

Our work is based on the following assumptions:

- Connection establishment (i.e. how devices discover each other), and other network related issues (like routing and roaming) are solved by the underlying system.
- We work with adaptive applications (streams), i.e., applications are capable to cope with changes in the underlying environment.
- Playout routes (paths) of all streams in the network are known. In other words, network topology, which describes a source and target device, likewise all other devices involved in one streams playout route, are considered to be known.
- Available resources (e.g., CPU, bandwidth) can be treated in isolation. We assume that there is no relationship between different

type of resources, and a certain QoS level has just one dimension. Hence, every quality level depicts resource supply/requiremets for just on type of resource.

- Applications (streams) descriptions in terms of quality levels, and their resource requirements are known.
- Synchronization between applications (streams) is left outside of our work.
- We assumed a linear mapping between frame (or bit) rate and quality of a stream, e.g., the higher frame rate the higher quality (see Section 2.2). The complex relationship between user perceived quality and resource demands is beyond the scope of the project.

1.3.2 Contribution 1: Real-time Resource Management Framework

From the beginning, resource management in home networks has been the focus of our interest. Efficient transport of streams with acceptable playout quality in heterogenous, dynamic environment (e.g. home networks) requires management of both networks and CPUs. We believe that key issues to enable resource handling with real-time methods in home networks are the interfacing between devices and resource management, providing a relevant view of the system state and diffusion of decisions to the devices. Therefore, when we designed our adaptive framework, the Matrix [39], we have attached great importance to finding tradeoffs between accuracy of system state information and efforts to transport and process it. Similarly, decoupling communication and synchronization between devices and resource management have played an important roll. The Matrix is based on a global abstraction of device states, which reduces system state information and decreases overheads for its determination and dissemination. It provides access to the entire system state in acceptable fresh way, enabling system wide optimized decisions to be taken. The Matrix is composed of several entities that

constitute an effective mechanism for monitoring and scheduling available resources in the system.

1.3.3 Contribution 2: Integrated QoS Adaptation Approach

We have extend the Matrix framework to implement a method for an efficient QoS provision and adaptation in dynamic, heterogeneous systems. As we already mentioned before, QoS adaptation can be performed *locally* on a device or *globally*, on the system level. While most of the existing approaches provide mechanisms for either local or global adaptation, we believe that both methods should be used together in order to respond properly to both local and global fluctuations. Hence, we propose the *integrated* global and local QoS adaptation mechanism, where the structural and long-term load variations on the system level are object for global adaptation, while the stochastic load and short-term resource variations are taken care locally on devices. The task of the local adaptation mechanism is to adjust resource usage locally on a device as long as the fluctuation is kept within a certain QoS range. If the resource usage exceeds the range's threshold, the global adaptation mechanism takes over and performs resource reallocation on the system level. In our approach, global adaptation (resource reallocation) is performed by a global resource manager, while the local adaptation is taken care of locally on the devices (by local monitors and schedulers).

QoS-aware applications are usually structured in such a way that they can provide different discrete quality levels, which have associated estimations of the required resources. We use the notion of abstract quality levels for defining QoS ranges, such as *high*, *medium* and *low* resource availability. This provides a general QoS framework that is not application or device specific. As long as an application or a device can express its resource demand and consumption in terms of an abstract level, it can benefit from our proposed solutions.

1.4 Outline of the thesis

The rest of the thesis is organized as follows:

Chapter 2 presents the Matrix framework for real-time resource management for video streaming in networks of heterogenous devices. The Matrix's ideas and its architecture are discussed here.

Chapter 3 describes the integrated Quality-of-Service adaptation approach for dynamic, heterogeneous systems. The integrated QoS approach is enabled by the Matrix framework, and we have applied it in the context of video streaming applications.

Chapter 4 contains analysis, results, and discussion of both, the Matrix framework and integrated QoS approach.

Chapter 5 concludes the thesis with a brief summary of the main contributions, and with an outline of future work alignments.

Appendix A gives an overview of the current status of the Matrix framework implementation, and also contains some ideas and discussions on the future implementation work.

Chapter 2

The Matrix: Real-time Resource Management Framework for Distributed, Heterogeneous Systems

In this chapter we propose an adaptive QoS framework for efficient resource management, called the Matrix. The Matrix is a concept to abstract from having detailed technical data at the middleware interface. Instead of having technical data referring to QoS parameters e.g., bandwidth, latency and delay, we only have discrete portions that refer to levels of quality. The underlying middleware must interpret these values and map them on technical relevant QoS parameters.

2.1 Rationale

In real-time systems, methods for scheduling and allocation have been developed to provide resource management including guarantees. These are capable of determining whether resources are sufficient for the timely completion of tasks and, if so, provide reservation methods to ensure that these resources are available for the guaranteed task when executing. Real-time resource management can thus manage all resources: it keeps track of the state of resources, provides admission control and resource guarantees. The resource management methods from real-time systems can provide key capabilities for the handling of streams and devices in home network.

However, the area of home networks introduces a number of crucial differences and issues compared to classical "process control" in real-time systems. For example, home networks are *heterogeneous*, where different local schedulers for CPUs and networks exist on diverse devices. An approach with tight coupling between global management and diverse local schedulers is not appropriate, since it will have unfeasible high overheads and require a fixed, known set of schedulers. Then, in home networks we have to cope with *limited resources* where real-time activities, including schedulers, execute on the same resource they are handling. Optimum scheduling can easily take more CPU cycles than the execution of the scheduled software takes. Finally, the resource demands coming from different applications are usually *highly fluctuating* over time (e.g., structural and stochastic load variations in video processing).

Key issues for enabling resource handling with real-time methods in home networks are efficient representation of the fluctuating system state, resource allocation decisions, and dissemination of orders. The overhead to transport the information needed for a 100% accurate view of the system with very fine grain granularity capturing highly fluctuating resources such as wireless networks will be prohibitively high on the network; scheduling activities for all events will overload CPUs. In addition, such information would be too fine-grained fluctuating, as re-

source management has to operate at larger granularity. We believe that the tradeoffs between accuracy of system state information and efforts to transport and process it have to focus on efficiency providing the minimum relevant information for resource management only.

Hop-by-hop approach

Efficiency of distributed resource management is critically threatened by overheads, as devices have to exchange information to determine a global system view for resource management decisions. In addition, scenarios such as high fluctuations on a network link demand more scheduling activities, which in turn will create more network overhead, resulting in increased fluctuations.

A hop-by-hop approach overcomes some of the problems of the fully distributed approach, such as network overhead. In the hop-by-hop approach decisions about how much of a resource is dedicated to a given version of a stream (application) are taken locally by the devices in sequence. However, it is impeded, by several shortcomings as well: the decisions taken on each device suffer from the limited view of the state of the device and the next one on the route. Suppose the resources on both devices can provide ample availability at the moment, resulting in the choice of a high quality/high bandwidth version of the stream to be transmitted. If any of the devices on the route to the play-out can handle only a lower quality version, the resources used here for the high quality will be wasted. Propagating the state information back and forth along the route results in the overhead described earlier and delays the actual scheduling in each device further. Rather, a sender-based approach with global knowledge is appropriate.

A further issue affected by the limited local state knowledge concerns the decomposition of end-to-end delay of the stream into deadlines for each of the devices on the route, forming deadlines for task scheduling and transmission delays.

Chosen approach

When designing our adaptive framework, the Matrix, we have attached

great importance to finding trade-offs between accuracy of system state information and efforts to transport and process it. The accuracy of the information represented is suitable for resource management, abstracting over fluctuations or changes, which will overload scheduling: the very fine grain resolution of values is mapped into a very small number or discrete values. Thus, the overhead to keep system wide state information fresh is dramatically reduced.

Moreover, in the Matrix we also provide an interface to decouple device scheduling and system resource allocation. Instead of the resource manager probing local schedulers for state information, the devices provide information about relevant state information and estimations about changes with appropriate, individual granularity themselves. Likewise, diffusion of decisions for resource allocation on individual devices, made by the resource manager, are carried out via the the Matrix as well. Thus the global resource management is independent of detailed knowledge about local schedulers, which can be replaced easily, supporting a component based, decoupled approach. Further, without the need for explicit costly communication and negotiation between devices, decision about which version of streams to transport or the decomposition of end-to-end delays can be performed by global resource management, reducing overheads and resource waste due to limited local device knowledge. Moreover, failures in devices or communication will not block or delay resource management.

2.2 QoS levels

The basic idea of the Matrix is to provide a global abstraction of device states as representation of the system state for resource management and to decouple device scheduling and system wide resource allocation. Further, in the Matrix we want to use the minimum relevant information about devices states as needed for resource management, in order to reduce the system state presentation, and to abstract over fluctuations, which could overload scheduling of resources. Thus, we use the notion of a few *abstract QoS levels* that represent a resource's availability and

an application's quality (see Figure 2.1).

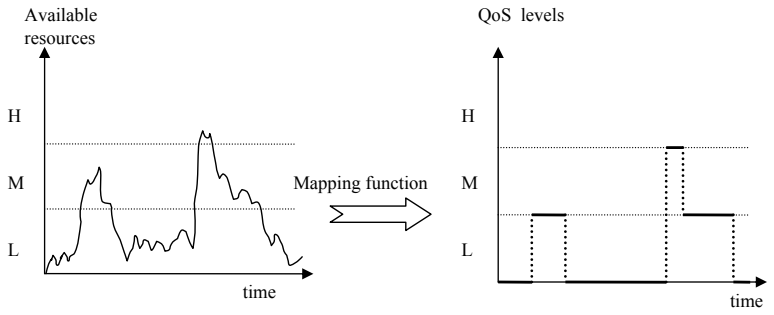

Figure 2.1: Reduced system state in the Matrix

For example, the variations in the quality of network link connection between two devices can be represented by e.g., three abstract QoS level values, (L)ow, (M)edium and (H)igh. H means that the data can be transmitted through the link with full available capacity, while L indicates severe bandwidth limitations. Likewise, quality of each application using certain resources is mapped to a finite number of application QoS levels.

In general, the availability of each resource is represented in our approach as a vector of discrete range of *n* QoS performance levels $\{q_1, q_2, ...q_k, q_{k+1}, ..., q_n\}$. The value range of a QoS level q_k is defined by its threshold values [q_k^{min}, q_k^{max}].

In this work, we apply linear mapping between the resources and the QoS levels, e.g., based on experimental measurements [26]. For example, one simple mapping for the CPU bandwidth based on the CPU utilization U could be e.g.,:

$$\begin{aligned} 0 \leq U \leq 0.3 &\Rightarrow H \\ 0.3 < U \leq 0.6 &\Rightarrow M \\ 0.6 < U \leq 1.0 &\Rightarrow L \end{aligned}$$

Likewise, we can map a video stream of a movie, to the following threshold values: 1 Mbps (L), 2 Mbps (M) and 3 Mbps (H). In that way, the number of state changes and associated resource management activities would be reduced drastically, while still providing a reasonably accurate representation of the actual stream demand (see Figure 2.2).

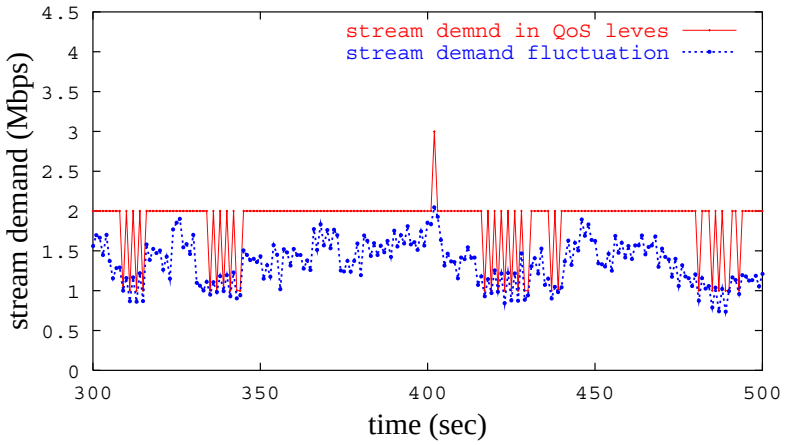

Figure 2.2: Mapping of a stream demand to few abstract QoS levels

However, having available resources mapped to just a few quality levels can also bring some negative effects. Thus, it implies an inevitable reduction of freshness system state presentation. Furthermore, in the case of video applications, too many quality changes can be preserved by a user as a significant quality degradation [8]. Again, it is very important to find a trade-off, between accuracy of the system state presentations, and efforts to transport and process.

A more advanced mapping could be, for instance, to use fuzzy logic to provide a larger number of QoS levels with finer granularity. However, QoS mapping is an ongoing work and it is out of the scope of this thesis. Still, it is very important to stress that both quality degradation, likewise quality promotion, should be done gracefully.

2.3 Architectural design aspects

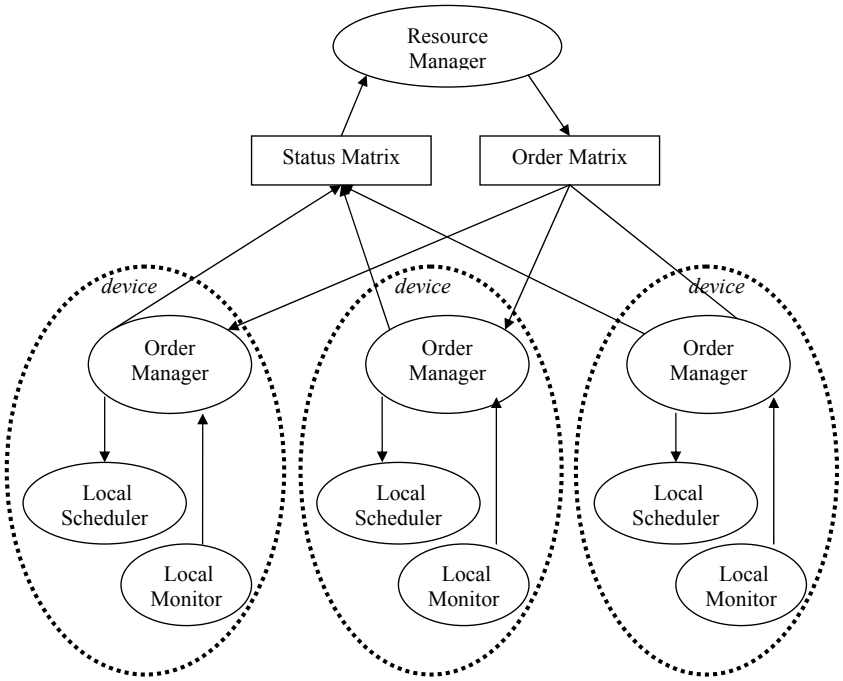

Figure 2.3: Information flow between Matrix's components

The Matrix framework is composed of several entities that constitute an effective mechanism for scheduling and monitoring of available resources in the system (see Figure 2.3).

In the following subsections, every part of the Matrix framework will be further described.

2.3.1 Resource Manager

For each "domain"[1] there is one *Resource Manager*. The *Resource Manager* is used to globally schedule and reserve resources in the system (domain), i.e., it makes decisions for resource usage for all devices. One part of stream scheduling is providing end-to-end timing constraints, i.e., providing sub deadlines (sub delays) for each device in the system by real-time methods. In other words, each time a resource variation occurs that affects an active video stream, the resource manager has to make an adjustment of streams and resources. Likewise, each time a new application is about to enter the system, the Resource Manager performs admission control.

In order to deal with resource reservation, the Resource Manager has to have knowledge about currently available resources in the system. This information is obtained from the Status Matrix. Based on that information, the Resource Manager will make decisions for resource reallocation in the system, and store the orders for devices in the *Order Matrix*. These orders can be seen as an interface between the resource manager's global view of resources and set of entities (order manager, local scheduler and local monitor), which we call "local enforcement mechanism".

2.3.2 Status Matrix

The *Status Matrix* contains information about available resources in the system, which is provided by the *Order Managers*, located on the devices. For each type of shared resources there is one Status Matrix, where every device has an entry, called a StatusMatrixCell. As mentioned above, available resources will be expressed with a few different QoS levels. This implies that we do not need to bother about updating the Matrix with each oscillation around a value. Instead, Order Mangers will update the Status Matrix when a change to a different quality level occurs for a significant amount of time.

[1] A subnetwork within which common characteristics are exhibited, common rules observed, and over which a distribution transparency is preserved

Furthermore, each resource is represented by its

1. current value (out of the limited number range)
2. current granularity, i.e., the time interval until which the current value is likely to not change, and
3. likelihood that 2) holds.

A single link on, e.g., wired switched Ethernet, will have a high granularity interval and high likelihood, whereas a wireless link in a mobile environment might result in small values for each. While accurate and correct predictions will not be possible, these values support better estimates for the decisions of the resource manager than very pessimistic values only. Should the granularity interval be less than is useful for resource management, the associated value for the device state can be assumed 0.

2.3.3 Order Matrix

The *Order Matrix* contains directions for resource reservations on the devices, made by the Resource Manager. Similarly to the Status Matrix, there is one Order Matrix for each type of resources, and each device is presented by one element in the Order Matrix, i.e., an OrderMatrixCell. Hence, devices pick their orders form the Order Matrix (or from their Cell in the Order Matrix) in form of:

1. delay (sub-delay)
2. value (out of the limited number range, QoS performance levels)

2.3.4 Order Manager

An *Order Manager* is responsible for allocating resources at a device. It maps global resource reservation constraints (orders), made by the Resource Manager, to the concrete scheduling specification for Local

Schedulers. Another task of the Order Manager is to provide the Status Matrix with information about locally available resources, in form of well defined QoS levels. There is an Order Manager for all type resources. The information about available resources is determined repeatedly, but not periodically by the order manager. The accuracy of the information depends on a chosen temporal granularity. Hence, an Order Manager is responsible for:

1. collecting information about the available resources at the devices
2. transforming various kind of traffic specification into a few QoS levels and providing the Status Matrix with them
3. allocating resources at devices and providing parameters to Local Schedulers.

2.3.5 Local Scheduler

A *Local Scheduler* is responsible for scheduling of local resources, e.g., a network packets scheduler that adjusts the packet sending rate according to available bandwidth. It is placed on a device and together with the Order Manager, it enforces local resource reservation. As mentioned before, the Order Manager provides parameters to the Local Scheduler.

2.3.6 Local Monitor

The information about available resources is provided to the Order Manager through *Local Monitors*. Thus, Local Monitors are responsible for continuous monitoring of a resource availability on a device, e.g., the available CPU or the network bandwidth. The accuracy of the information depends on a chosen temporal granularity of monitoring intervals.

2.3.7 Operational scenario

Local Monitors on devices, monitor available resources continuously and send this information to Order Managers. The Order Manager per-

forms resource calculation locally to estimate the maximal available resources it can offer. Then, it maps these values to the appropriate QoS levels, and store them into the Status Matrix. Based on the information stored in the Status Matrix, the Resource Manager will make decisions for resource reallocation in the system, and store the orders for devices in the Order Matrix. Then, an Order Manager receives orders from the Order Matrix and makes sure to adjust local resource usage according to them. This is done through the Local Schedulers.

2.4 Communication between devices

In traditional computer world, most of interaction is performed in client-server model, where a client process requests and waits for some services (data) from a server process. Client and server are tightly coupled, i.e., , messages can not be exchanged unless both client and server are up and running, and fully aware of each others existence. However, we believe that for video and audio streaming application in home networks, likewise for dynamic mobile environment, a better solution for communication is a publish/subscribe model [12, 16]. In such model, a device, which acts as the publisher, makes some information available to the rest of the devices (i.e., a device publishes information), and has also the responsibility to update that information. Any other device interested in that information has to subscribe to it. The subscriber device receive only messages that it has subscribed to, without any knowledge of who the publisher is. Thus, one of characteristics of publish/subscribe system is its anonymity. Then, it is also asynchronous, since a sender (publisher) does not have to wait for acknowledgment from a receiver, but reliability of communication is taken care by the underlying infrastructure. Lastly, publish/subscribe is quite similar to multicast communication, where a sender (publisher) may send one message (information) to many receivers (subscribers), with just one publish operation.

In the Matrix approach we want to reduce the number of control messages between the Resource Manager and the devices (Order Managers), and by using publish/subscribe communication model, messages

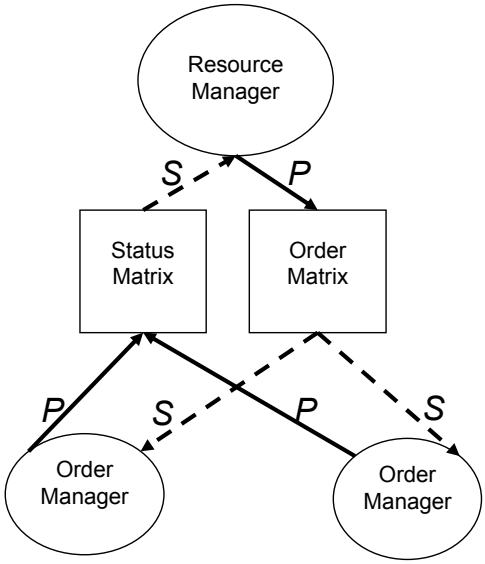

Figure 2.4: Matrix and publish/subscribe communication model

are only sent when attributes of the matrix elements (Status and Order Matrix) are actually changed. Moreover, decoupling of publisher and subscriber devices enables scalability and mobility in a network, which fits well within the Matrix.

Figure 2.4 describes how publish/subscribe communication model is adopted in the Matrix framework. As we mentioned before, resource availability information stored in the Status Matrix, is published by the Order Managers. Each Order Manager adds an entry in the Status Matrix and becomes owner of this object, which also means that it is responsible for updating it. The Resource Manager is interested in information published in the Status Matrix, thus it subscribes to the Status Matrix. Likewise, information in the Order Matrix is updated by the Resource Manager, i.e. the Resource Manager publishes elements of the Order Matrix. The subscribed Order Managers are reflected ac-

cordingly. So far, we have looked into two implementation of publish/subsrcibe mechanisms, HLA and xmlBluster [1, 2], and their applicability within the Matrix framework. We have found them both suitable for the Matrix framework (see Appendix A).

However, it is not necessary to use any of existing publish/subscribe mechanism with the Matrix framework. The parts of the Matrix, which otherwise directly benefit from from the publish/subscribe model, could be implemented by means of common programming methods. For example the communication parts of the Matrix framework could be implemented by pure sockets. But, the amount of the required implementation work would increase, in order to achieve anonymity and asynchronicity, which are offered by any publish/subscribe mechanism.

Finally, want to stress, a publish/subscribe mechanism is only considered for the management part of the Matrix framework. Possibilities of using it for the transport of streams have not been examined so far.

2.5 Motivating example

In order to give a more detailed description of how the different part of the Matrix framework work together, let us consider the following motivating example:

A person uses a PDA (Personal Digital Assistant) to watch a video stored on a local video server, which is delivered to the PDA through a wireless network. As the person moves around with the PDA, the amount of the available bandwidth varies, which can result in video interruption due to packet lost. Fortunately, the video server is able to provide the stream with different qualities, and the user can continuously watch the stream on his/her PDA, although with the varied quality.

For the sake of simplicity, in this scenario we consider just one type of resource, available network bandwidth. We assume the scenario network like one presented in Figure 2.5, where connection between the PDA and the video server is archived via one wireless router. Hence,

in our example scenario we have two connection links, A and B. Furthermore, in this scenario we omit the steps that describe the system and stream setup (for example, connection to the network, which lets the home network and the PDA discover each other), since those issues are not in the scope of our work, and we assume that they are taken care by the underlying system.

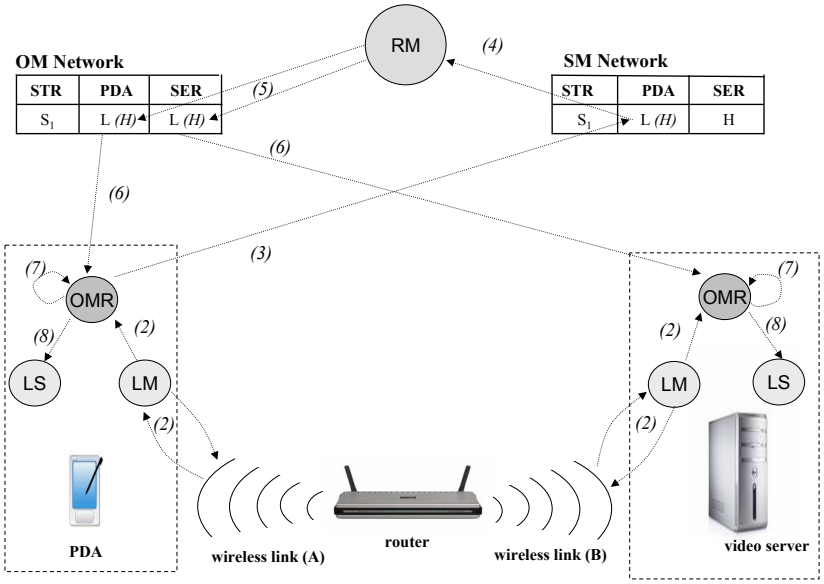

Figure 2.5: Scenario network

The steps of the scenario are following:

1. Each order manager, on devices, adds one or more entries, StatusMatrixCells, in the Status Matrix and becomes owner of these objects. If a certain device is also a source of streams, a new "row" in the Status Matrix, i.e., a set of StatusMatrixCells, will be created for each stream. (This step is omitted in Figure 2.5.)

2. The Local (Bandwidth) Monitors on the PDA and video server ob-

serve the current bandwidth value of the wireless network. Then, those values are being reported to the corresponding Order Managers.

3. The Order Manager compare the observed bandwidth value with the threshold values imposed by current quality level. So let us assume that the Order Manger on the PDA will discover that the currently observed quality level by the local monitor on the PDA (on connection link A) is different from the previously published one. In this case, the Order Manager will publish this new QoS in the Status Matrix.
   On the contrary, if the observed quality level would be the same as the previous one, there will be no need to publish it again.

4. The Resource Manager has subscribed to the elements in the Status Matrix and will only receive information when attributes of the elements are updated. (That is enabled by publish/subscribe communication mechanism that we have adopted in our framework, see 2.4). Thus, the Resource Manager is notified after publishing the new quality value for connection link A in the Status Matrix.

5. The Resource Manager, who has an overview of the whole resource situation (i.e., for all involved devices), maps available resources to the resource requirements. If there are enough resources to support the requested connection, the resource manager puts directions for resource reservation in the Order Matrix, by updating attributes of the Order Matrix elements.

6. Order managers subscribe to the elements in the OrderMatrix, i.e., OrderMatrixCells. Thus, when attributes in the OrderMatrixCells are updated, they become visible for Order Managers on all devices (again due to publish/subscribe mechanism).

7. The Order Managers on respective devices will try to make local resource reservation, i.e., to translate the abstract levels to application specific concrete values to be used by the Local Schedulers.

8. The Local Scheduler gets scheduling specification (bandwidth, delay) for each stream-part from the order manager, and adjusts the transmitted packets rate according to these new values.

2.6 Admission control

Each time a new application (video stream) is about to enter the system, the Resource Manager has to determine if sufficient resources are available to satisfy the desired QoS of the new connection, i.e. it has to perform an admission control. Admission control is initiated first by an user and goes via Order Manager on the device. Thus, the Order Manager updates the Resource Manager (via the Status Matrix) with application's info (e.g., resource requirements), and the Resource Manager makes decision. If one application can be accepted without violating the QoS of existing applications, the Resource Manager will simply publish new orders for resource reservation/reallocation into the Order Matrix. However, in the case of insufficient resources, different steps could be taken depending on current resource reallocation policy. Thus, if applications are not assigned any priorities, we simply reject the new application. On the other hand, when priorities have been used, we have to check if there are any existing application with lower priority than the new one, and if so decrease their quality levels (QoS), in order to free some resources for the new application. If available resources are insufficient, despite maximum allowed QoS degradations of the lower priorities applications, the new application is rejected. In section 3.4, we will describe the reallocations policies that we have implemented in our work.

2.6.1 Control messages during admission control

We can measure the cost of admission control, in terms of control messages that have to be exchanged within the system.

When a user wants to start one new application, it goes via the Order Manager on the device. Thus, the Order Manager sends (publishes)

an update message to the Resource Manager (via Status Matrix). Depending if the Resource Manger rejects or accepts this new application, it will send message(s) either to the calling Order Manager, or to all Order Managers on involved devices. Figure 2.6 describes how control messages are sent during an admission control phase. (The arrows with dash line style denote the messages that are only sent when admission control accepts the new application).

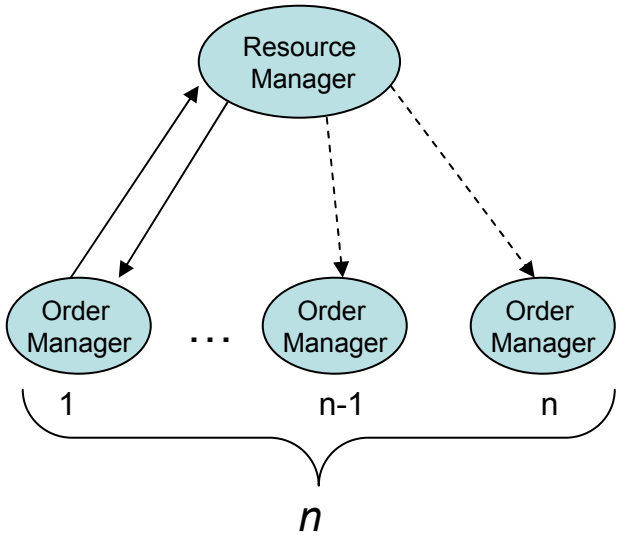

Figure 2.6: The Matrix: the control messages during an admission control.

Let n denote the number of devices in the system, and $\mathcal{M}$ the number of control messages that has to be sent for an admission control. Then, we have:

$\mathcal{M} = 2$,

when a new application is rejected. In this case, besides the first

message which is sent by the Order Manager to the Resource Manager, just one additional reject message is sent back from the Resource Manager to that Order Manager. The rest of the devices (Order Managers) are not involved in this process, since their resource status is not changed.

However, when available resources are sufficient, i.e., when an application is accepted, $\mathcal{M}$ is equal to:

$$\mathcal{M} = n + 1,$$

because, the Resource Manager has to send messages to all involved Order Managers on the devices.

Hence, a rejection of a new application within admission control costs just two control messages. On the other hand, the amount of control messages in case of an application's acceptance is proportional to the number of devices in the system, i.e., it increase/decrease with the increased/decreased number of devices.

2.6.2 Comparison with hop-by-hop approach

During an admission control in a hop-by-hop approach, control (resource allocation) messages have to be sent from a source to the sink device, and back (see Figure 2.7). Consequently, in one hop-by-hop approach the amount of control messages during an admission control phase, is:

$$\mathcal{M} = 2(n - 1)$$

However, this equation for $\mathcal{M}$ is valid when the new application is accepted, or rejected by the last device on the path (the sink device).

When a first (source) node discovers that the new application can not be accommodated, then $\mathcal{M} = 0$. Thus, in cases of an application's rejection, the number of control messages, $\mathcal{M}$, can take any value in the

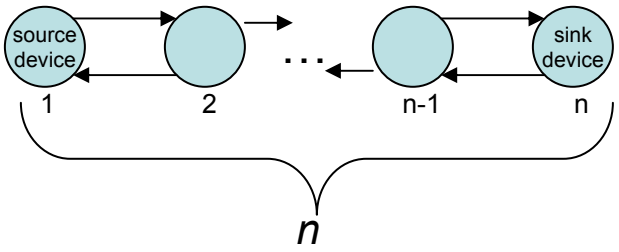

Figure 2.7: Hop-by-hop approach: the control messages during an admission control

interval $[0, 2, 4, ..., 2(n-1)]$, depending on which node discovers a lack of available resources.

Finally, we express the number of control messages during an admission control in a hop-by-hop approach, as follows:

$$\mathcal{M} \leq 2(n-1)$$

Figure 2.8 shows the relations between the number of devices in the system, and the cost for admission control in terms of control messages, for both the Matrix and hop-by-hop approach. We can see that a rejection of an application during admission control always costs two control messages in the Matrix, while in the hop-by-hop approach this vary from zero to 2(n-1). On the other hand, in both approaches, there is a proportional dependency between the amount of control messages and the number of devices (nodes), in cases when an application is accepted. Clearly, the more devices the more control messages are needed to establish a new application in the system. However, admission control in the Matrix generally requires less control messages when the number of devices (nodes) in the system is greater than three, which is quite

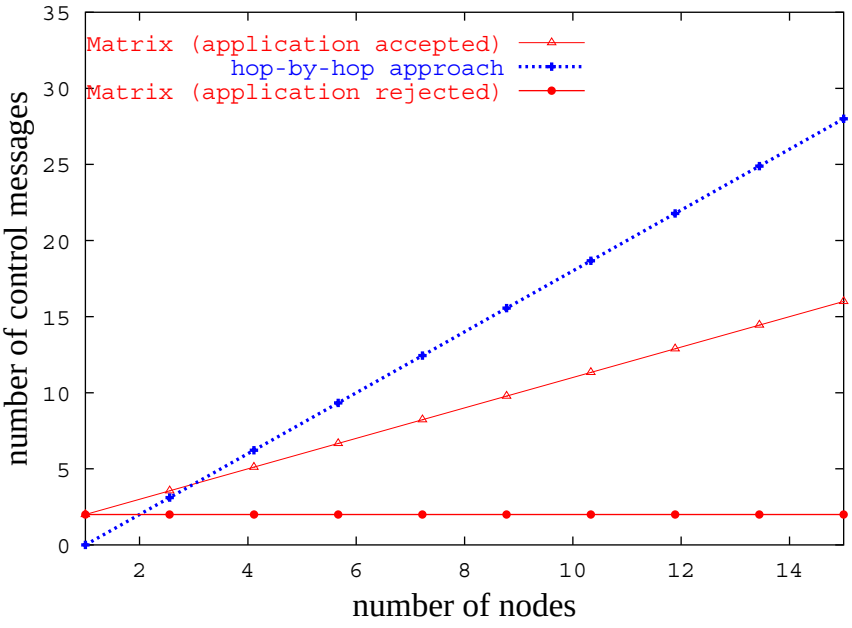

Figure 2.8: Control messages during admission control; the Matrix vs hop-by-hop approach.

common situations in home networks environments.

2.7 Resource reallocation delay

Delay is a term often mentioned in communication literature, as the time required for a message to travel from the transmission point to destination. Concretely, in our work we are talking about resource reallocation delay, which is a time lag between the moment an device announces (publishes) a change in resource availability, and the moment when all devices get new resource reallocation orders.

In the Matrix framework a change in resource availability on a device is published by an Order Manager to the Status Matrix. Then,

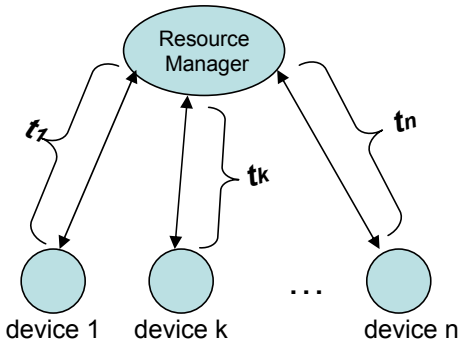

Figure 2.9: A resource reallocation delay in the Matrix

the Resource Manager makes resource allocation decisions which are spread (broadcasted) to the initiating device, plus all other involved devices. As devices are connected over different types of networks (we work in distributed heterogeneous environment), a message transmission time, between two devices in the system, may vary quite a lot. In other words, the time it takes to send one control message from the Order Manager on a device, to the Resource Manager, is not the same for all devices in the system.

For this purpose, let $\mathcal{T}$ denote a set of all possible worst case message transmission times between two devices in the system (or between Order Managers and Resource Manager):

$$\mathcal{T} = \{t_1, ..., t_n\},$$

where n denotes the number of devices in the system. Obviously, in a home network environment many connection links may be similar, thus many elements in $\mathcal{T}$ may have the same value.

Let t_k denote a worst case message transmission time on the link be-

tween the device k, who has initiated a resource reallocation, and the Resource Manager. Thus, t_k is one element in $\mathcal{T}$, $t_k \in \mathcal{T}$.
For instance, t_k on a wired switched Ethernet link, will have a low neglected value (≈ 0), whereas t_k on a wireless link in a mobile environment might result in higher values.
We also introduce t_{max} as the maximum message transmission time in $\mathcal{T}$:

$$t_{max} = \max \{t_1, ..., t_n\}$$

Hence, we say that the resource reallocation delay, in the Matrix, is :

$$\mathcal{D} = t_k + \mathcal{C} + t_{max} ,$$

where $\mathcal{C}$ is a worst case execution time of the Resource Manager task. One special case is when $t_k = t_{max}$. Then, the upper limit of $\mathcal{D}$ can be calculated as:

$$\mathcal{D} = 2\, t_{max} + \mathcal{C}$$

On contrary, in the case that all links are wired, we approximate to the low limit of $\mathcal{D}$, i.e.,

$$\mathcal{D} = C + 2t_k \approx C.$$

At the end, we can say that a resource reallocation delay in the Matrix is:

$$\mathcal{D} = t_k + \mathcal{C} + t_{max} \leq 2t_{max} + \mathcal{C}$$

Consequently, a resource reallocation delay in the Matrix is independent of the number of devices (nodes) present it the system, i.e., a larger number of devices in the system does not imply a large delay. We

want to stress that this delay calculation is also valid for the admission control in the Matrix, as the resource reallocation mechanism is same.

2.7.1 Comparison with hop-by-hop approach

As we mentioned above, a resource reallocation delay in the Matrix is independent of the number of devices in the system. On the other hand, in the case of hop-by-hop approach a resource reallocation delay (e.g., admission control) increases with the number of devices.

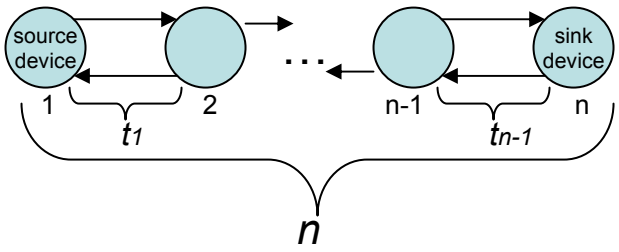

Figure 2.10: A resource reallocation delay in a hop-by-hop approach

Let, n be the number of devices one control message has to pass on its way from the source to the sink device (see Figure 2.10). Also, if C^{tot} denotes the total time (WCET) spent by all devices in order to estimate available resources, i.e.,

$$C^{tot} = \sum_{k=1}^{n} C_k.$$

Then, a resource reallocation delay can be calculated as:

$$\mathcal{D} = 2\sum_{k=1}^{n-1} t_k + C^{tot}$$

In the worst case scenario, i.e., when all t_k are equal to t_{max}, the upper bound of $\mathcal{D}$ is:

$$\mathcal{D} = 2(n-1)t_{max} + C^{tot}$$

Finally, we say that in the hop-by-hop approach a reallocation delay is:

$$\mathcal{D} = 2\sum_{k=1}^{n-1} t_k + C^{tot} \leq 2(n-1)t_{max} + C^{tot}$$

Accordingly, in the hop-by-hop approach, a resource reallocation delay depends on the number of devices, i.e., it increases with the number of devices in the system, and vice versa. It is obvious that the number of devices has an impact on the amount of control messages that have to be sent between devices, but also C^{tot}, depends on how many devices have been involved in the resource reallocation process. The more devices in the system, the more time is totally spent on resource reallocation.

2.8 Chapter summary

In this chapter, we have presented the Matrix, an adaptive framework for applying real-time resource management methods for decoupled video streaming of heterogenous devices. The Matrix is based on a global abstraction of device states, which reduces system state information and decreases overheads for its determination and dissemination. Instead of having technical data referring to QoS parameters, e.g., bandwidth, latency or delay, we only have discrete portions that refer to levels of

quality. Also, the Matrix is an efficient system state representation and an interface to decouple device scheduling and system resource allocation, thus enabling resource reallocation without explicit consideration of intricate details of device schedulers.

Concurrently, we want to stress that the Matrix provides a logical abstraction of the view of the system state, not an actual centralized implementation requirement. Rather, the Matrix is represented in a distributed way.

Theoretically, it is possible that a single device does not participate in the Matrix approach. However, the higher abstraction level provided by the Matrix and decreased overhead would be lost. Obviously, the resulting complexity would effect the quality of the decisions and the overheads as well as impact the rest of the system. Devices either join the Matrix fully, i.e., provide input to the Status Matrix and follow the Order Matrix, or not at all.

Moreover, we have compared the Matrix to th hop-by-hop approach, with respect to admission control and resource reallocation delay.

Chapter 3

Integrated QoS Adaptation Approach

In this chapter we present a method for efficient Quality-of-Service adaptation in dynamic, heterogenous environments. Implementation of the proposed integrated QoS mechanism is enabled by the Matrix framework, which makes possible adjustment of load fluctuations on different architectural levels. We propose a local adaptation mechanism on system devices to deal with short-term resource fluctuations, and combine it with the global resource management of Matrix that handles long-term load variations.

Moreover, we introduce an application adapter, the closed-loop control model for resource monitoring and adaptation, the different resource reallocation policies used within global adaptation, as well as the deployment of our approach in the context of video streaming and video stream adaptation.

3.1 Integrated QoS Adaptation and the Matrix

In this section we present our integrated global and local adaptation mechanism that uses the Matrix framework. In our approach, global adaptation (resource reallocation) is performed by the Resource Manager, while the local adaptation is taken care of locally on the devices.

Let us look again on the first part of the motivating example presented in 2.5:

A person uses a PDA to watch a video stored on a local video server, which is delivered to the PDA through a wireless network. As the person moves around with the PDA, at some point it becomes almost out of range for the server, which results in video interruption due to packet losses.

A local adaptation on the PDA does not really help in this case, since the video disruption is caused by the buffer underflow in PDAs decoder (in the case of buffer overflow, this could be treated locally on the PDA by i.g., speeding up the video decoding task). However, if there is a mechanism at the system level that can detect the lower bandwidth of the wireless link, i.e., the Matrix framework described in previous chapter, it could instruct the video server to stream a lower quality video stream that requires less network bandwidth.

Expressed in more general terms, resource consumption is adjusted locally on devices as long as the fluctuation stays within the range of requested QoS. For example, the Local Monitor detects a change in available CPU for a certain application, but this change is not large enough to enforce a different quality level to the application. Instead, the Local Scheduler could perform some local countermeasures, e.g., prioritize the application on the cost of some other application running on the same device. However, if the resource availability passes the defined thresholds (abstract QoS levels), the entire system gets involved via the global adaptation mechanism. The whole idea is illustrated in Figure 3.1.

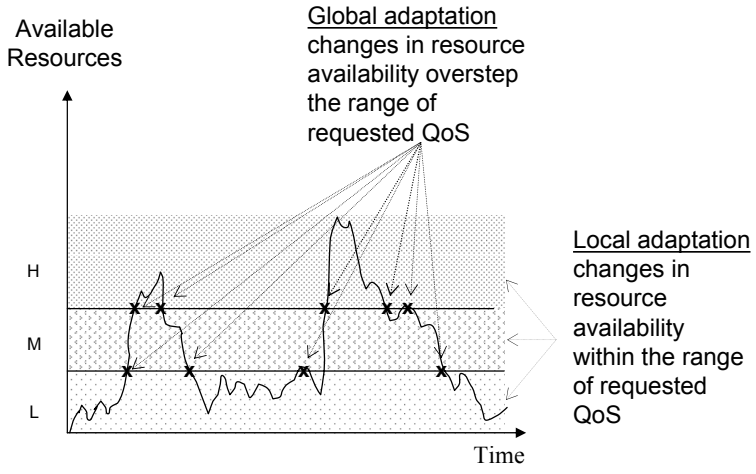

Figure 3.1: Different types of resource variations handled on different architectural levels

3.2 Application Adapter

The Matrix is an application independent framework, and application adaptation is not the main focus of our work. However, in order to advance the usage of the Matrix along with various types of applications, we have extended the original Matrix framework with an additional component, the *Application Adapter* (AA). The Application Adapter performs the mapping of QoS levels to the application specific parameters, and vice versa. For example, the AA ,for a video streaming application, could map abstract quality levels, such as H, M and L, into real possible frame-per-second (fps) values for the stream. Thus, for a 30 fps MPEG-2 stream high quality could mean the fps-interval between 24 and 30 fps, medium quality is 16 to 23 fps and low quality could be defined as 10 to 15 fps.

Since this process is application specific, our ambition was to provide an interface for this component, and then is up to the application

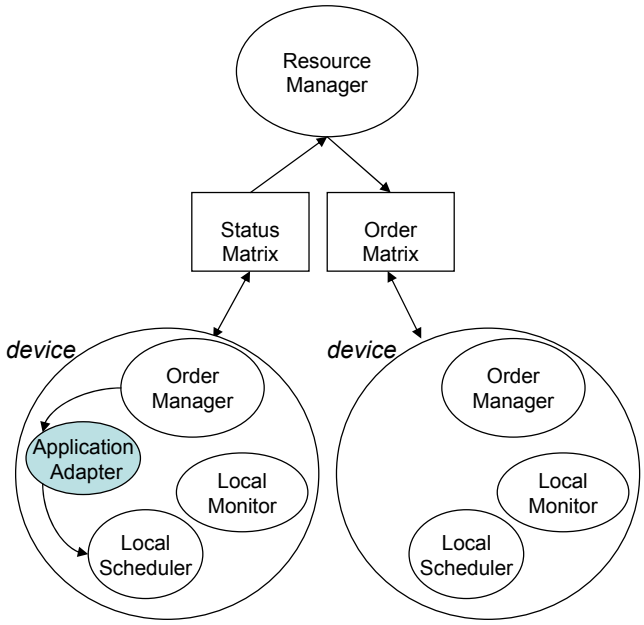

Figure 3.2: Application Adapter and the Matrix framework

designer to implement it. If there is a way in an application to map its resource fluctuations into some abstract levels, then it can be used with our design. At the same time, upon resource reallocation, the Application Adapter will receive orders about new abstract levels from the Order Manager which must be translated into some concrete actions on the application level.

Accordingly, the Application Adapter interfaces with both Order Managers and Local Schedulers. After performing the adjustment of one application, the Application Adapter sends the data to the Local Scheduler. Figure 3.2 depicts a placement of an Application Adapter in the Matrix framework.

3.3 Local adaptation mechanism

Local adaptation involves detecting the changes in resource availability and reacting to those via some local mechanism. The ideas from control theory can be used to achieve this. We use the *closed loop* model, i.e., a control model that involves feedback to ensure that a set of conditions is met. It involves the Local Monitor, the Local Scheduler, and the Order Manager (see Figure3.3). Expressed by terminology of the control theory, we use the following terms for input and output variables in our control model:

- *control variable*, v_{ctrl}, is the value observed by the Local Monitor (e.g. network packet loss, CPU utilization).
- *reference variable*, v_{ref}, is concrete performance specification for Local Schedulers made by the Order Manager.
- *error* ε is the difference between the control variable and the reference variable.
- *control input variable*, v_{in}, is the value calculated by the adaptation algorithm in order to adapt scheduling of the local resources.

The Local Monitor continuously monitors available resources in the system (e.g., CPU or bandwidth). Thus, in our control model, it acts as an *observer* of the controlled system. It sends the observed control value to the Order Manager. The Order Manager calculates the difference between the desired value, as defined by the currently used QoS level, and the observed control value, i.e., it calculates the error value of the control loop. As long as the resource availability stays within the boundaries for the given QoS level, i.e., the error falls in the range of the current QoS level, the output of the adaptation algorithm, control input, is passed to the Local Scheduler, i.e., the adapter part of control loop.

In the case the error value indicates a change in QoS levels, the values in the Status Matrix are updated and the Resource Manager is informed about the change. From this point, the global adaptation mechanism takes over, which we describe next.

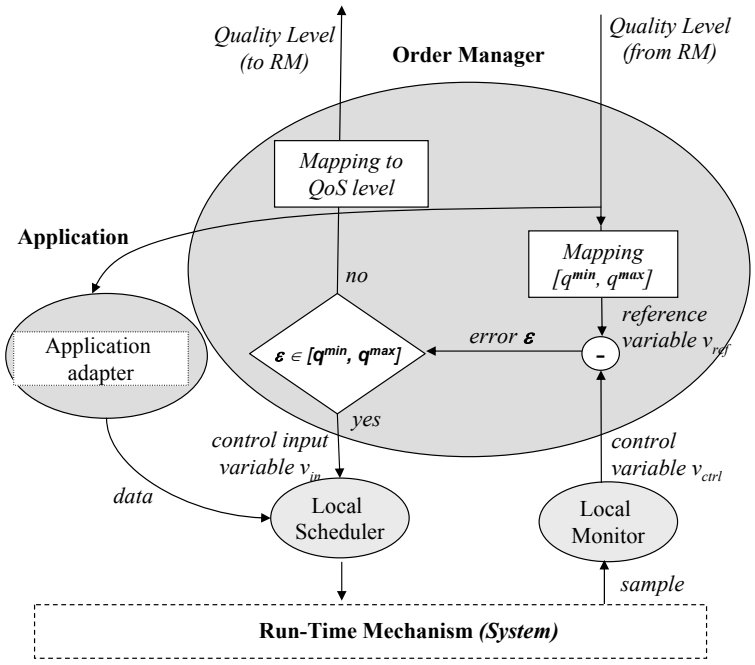

Figure 3.3: Local QoS Adaptation Mechanism

3.4 Global adaptation mechanism

Whenever a local mechanism detects that a local resource availability has exceeded the current QoS level, a global adaptation mechanism will be initiated. The objective of the global adaptation is to adjust the resource usage among all involved applications. If the resource availability has increased, it will be distributed among the applications (in terms of increased quality levels). Similarly, if the resource availability has decreased, the quality levels of the consumer applications will be decreased.

We support user defined *priorities* between applications to be used

when redistributing resources, i.e., the higher the priority of an application, the faster the quality increase it gets. However, it is up to the user to use priorities or not. Based on this, we distinguish between four reallocation policies in our approach, *naive, fair, fair-prioritized, and greedy*.

Naive reallocation The very first resource reallocation method that we have implemented in the Matrix framework was a naive method. It is very simply in its nature. In this method, applications are not assigned priorities, and we adjust applications and available resources quality levels according to the minimum quality level among them. Although, this method maybe does not use the available resources in the best possible way, and degrade slightly overall quality of applications, its main advantage is simplicity.

Fair reallocation If the priorities are not used, then the resources are adjusted (increased or decreased) in a strictly fair fashion: for each consumer applications the quality is adjusted step-by-step, one QoS level at the time, and then, if there are still resources to increase/decrease, we repeat the procedure for all applications once again, until the resource is consumed/replanished. For example, consider four different applications a_1,a_2, a_3 and a_4 that are using the same resource r. The current quality level for each applications is set to L. Assume that a_4 gets terminated and the resource availability of r gets increased by the portion used by a_4. The freed resource is given back to the remaining three application such that we first increase the the QoS level of a_1,a_2 and a_3 to M, and then, if there are still resources left, all QoS levels are increased to H.

Fair-prioritized reallocation Note that in the fair approach, there is no guarantee that a certain application will change its QoS level. In the example above, there could be a case where the freed resource is entirely consumed after increasing the level of a_1 and a_2 to level M, so that a_3 will remain running on level L, despite the fact that a_3 might

be the most important one in the system. However, if we use priorities, we could instruct the Resource Manager to start by increasing the QoS levels of high priority applications first, i.e., a_3 in the example above. In other words, the resources are reallocated in a fair fashion, i.e., each application's quality level is changed by one step before changing any other application's level one more step, but also we use priorities to determine which applications should be served first.

Greedy reallocation Moreover, priorities enable for an another reallocation policy, i.e., greedy redistribution. This means to increase (decrease) QoS level of an application with the highest (lowest) priority until it reaches its maximum (minimum) QoS level, before we start with the next application (in the priority order). For the example above, we would continue increasing the QoS level of a_3 until it reaches H, before doing any QoS increase of a_1 and a_2. Furthermore, the priorities can be used when selecting which applications to drop first if that becomes necessary.

If an application is processed by several different devices, then, before changing its quality level, we need to check if the new level can be supported by all involved devices on the application's playout route. For example, in a video streaming application where a video stream is sent from a video server to a hand held device via a laptop, the bandwidth increase between the server and the laptop does not necessarily mean that we should start streaming a higher bit rate stream, since the link between the laptop and the hand held device might not be able to support it. Likewise, we have to consider if this increased quality can be supported by all other types of resources that the application is consuming e.g., there is no point to send more data over the communication link than it cannot be timely processed at the receiver device (by the local CPU).

At the end, we increase the quality level of the application and if we still have spare resource capacity, we continue with the next application, based on the chosen adaptation policy.

If the available resources in the system are decreased, then the QoS for some applications needs to be degraded. We always start with the lowest priority application and decrease its QoS to the next (lower) QoS level. Then we reallocate the freed resources gained by this quality degradation. Again, in our current implementation, in this step we just consider one resource type (the one that we start the adaptation with), without considering how it will influence other available resources. Then, if the resources are still insufficient, we decrease the QoS level of the next application, based on the chosen adaptation policy, and so on. Our policy is to lower the QoS of all active applications before dropping some of them.

In our current implementation we only check if we have enough available resources to mach this new quality of the application, and do not consider further reallocation of resources. However, a more complex algorithm could take into consideration that the changed (increased or decreased) resource utilization of one resource type may result in the changed resource status for other resource types (e.g. in video streaming application, more available bandwidth on the communication link, could mean less video encoding on the server side, thus less usage of CPU resources). In that way, we could free immediately some other resources, without waiting for local monitors to discover this foreseeable change in the resources availability. This resource usage dependency is completely application specific. Hence, an Application Adapter could provide this information, obtained offline, e.g. by some test runs, at the start up to an Order Manager. Then, the Order Manger could publish this information to the Resource Manager (via the Status Matrix).

3.5 Pseudo-code for the Integrated QoS Approach

In this section, the pseudo-code for our current implementation of the integrated local and global QoS adaptation mechanism is presented. We introduce some additional terms, as a complement to the terms presented

earlier:

- $\mathcal{A}$ = $\{a_1, a_2, .., a_n\}$, the set of applications in the system.
- $\mathcal{R}$ = $\{r_1, r_2, ..., r_m\}$, the set of resources in the system.
- $\mathcal{D}$ = $\{d_1, d_2, ..., d_p\}$, the set of devices in the system.
- $\mathcal{A}(r_i) \in \mathcal{A}$, the subset of applications that currently use resource r_i.
- $\mathcal{R}(a_j) \in \mathcal{R}$, the subset of resources currently used by application a_j.
- $\mathcal{R}(d_l) \in \mathcal{R}$, the subset of resources currently consumed on device d_l.
- $\mathcal{D}(a_j) \in \mathcal{D}$, the subset of devices currently used for processing of application a_j.
- $S(r_i)$, current resource supply (availability) of resource r_i.
- $D(r_i)$, current resource demand of all applications using r_i.
- $q_k(r_i)$ and $q_k(a_j)$, the *k*-th QoS level of resource r_i, respective application a_j, as described in section 2.2.

/ For the sake of readability, we omit the pseudo-code for the start up activities where the devices have reported the local resource availability, and the RM has published initial QoS levels in the Status Matrix */*

$\forall\ d_i \in \mathcal{D}$ ** For each device */*
  $\forall\ r_i \in \mathcal{R}(d_i)$ ** For each resource on a device */*

/ Invoke local adaptation based on the currently assigned quality level */*

map $q_k(r_i) \Rightarrow$[$q_k^{min}(r_i)$,$q_k^{max}(r_i)$]
$v_{ref} = q_k^{max}(r_i)$
$\varepsilon^{max} = q_k^{max}(r_i) - q_k^{min}(r_i)$
Do
get v_{ctrl} from LM
$\varepsilon = v_{ref}$ - v_{ctrl}
calculate $v_{in}(\varepsilon)$ and send it to LS
While *(*$0 \leq \varepsilon \leq \varepsilon^{max}$*)*

/ Prepare for global adaptation when the error exceeds the limit of the currently assigned quality level */*
map $\varepsilon \Rightarrow q_l(r_i)$, $l \neq k$
publish $q_l(r_i)$ in SM
$\Rightarrow$ *break! invoke global adaptation*

/ RM performs global adaptation based on new info in SM */*

/ Check if it is the naive policy*/*
If ($POLICY == naive$) ***Then***
$\forall\, r_i \in \mathcal{R}(d_i)$ */* For each resource on a device */*
$\forall\, d_i \in \mathcal{D}$ */* For each device */*
find the minimum QoS level *min*
$\forall\, d_i \in \mathcal{D}$ */* For each device */*
set $q_k(r_i) =$ *min*
Else*/*POLICY is fair, fair-prioritized, or greedy*/*
/ Case 1: total resource supply is greater than the total demand $\Rightarrow$ increase QoS levels */*
If ($S(r_i) > D(r_i)$) ***Then***
Do
/ Based on the chosen reallocation policy, get an application to increase its QoS level */*
If(a_j = getApplication(POLICY, INCREASE)) ***Then***

/ Check if all a_j's processing devices (other than d_i,*

which initated global adaptation) support
the next (increased) QoS level of a_j/*
If($\forall d_j \in \mathcal{D}(a_j), d_j \neq d_i, d_j$ supports $q_{k+1}(a_j)$) ***Then***

/ Check if the new QoS level of a_j can be served*
by all other resources that a_j consumes/*
If($\forall r_n \in \mathcal{R}(a_j), r_n \neq r_i, r_n$ supports $q_{k+1}(a_j)$) ***Then***
increase quality of a_j to $q_{k+1}(a_j)$
/ increase/decrease demand/supply for r_i by the*
*amount used to jump to the next quality level */*
$\Delta = q_{k+1}^{max}(r_i) - q_k^{max}(r_i)$
$D(r_i) += \Delta$; $S(r_i) -= \Delta$

While *($S(r_i) > D(r_i)$ AND $a_j \neq$ NULL)*

/ Case 2: total resource supply is less than*
*the total demand $\Rightarrow$ decrease QoS levels */*
Else
/ Similar as above, but the QoS levels are decreased*
Also, we do not need to check the other devices
and resources, since the decreased quality will not
*put extra demands on them, as in the case 1 */*
Do
If (a_j = getApplication(POLICY, DECREASE)) ***Then***
decrease quality of a_j to $q_{k-1}(a_j)$
/ decrease/increase demand/supply for r_i by the*
*amount used to jump to the lower quality level */*
$\Delta = q_k^{max}(r_i) - q_{k-1}^{max}(r_i)$
$D(r_i) -= \Delta$; $S(r_i) += \Delta$

While *($S(r_i) < D(r_i)$ AND $a_j \neq$ NULL)*

3.6 Example

In this section, we illustrate our approach in the context of video streaming. We continue to elaborate with the example scenario from 2.5 and 3. Thus, in our scenario, the quality of the streamed video was dependent on the distance between the PDA and the server. At some point in time, the PDA is so far away from the server so it only makes sense to stream a low quality video stream, i.e., stream S_1 with the abstract quality level L and priority p_1. Assume also that there is another video stream in the system, S_2, streamed from the server to a laptop with a quality level H and higher priority p_2. The CPU availability (bandwidth) on all devices is initially assumed to be high. The reallocation policy used is fair-prioritized. The whole situation is depicted in Figure 3.4. The values within the parentheses are the new QoS levels (obtained after adaptation).

Now, assume that the person with the PDA starts moving closer to the server. The local adaptation mechanism on either the server or the PDA will detect that more and more packets can be sent between them (let's assume the PDA will detect this first). As the PDA is coming closer to the server, at some point, the quality of the link connection will exceed the assigned threshold for the local adaptation, and the global adaptation mechanism will take over, with the following steps involved (see Figure 3.4 in parallel; the numbers below correspond to the numbers in the figure; some of the steps are merged):

1. The Local Monitor on the PDA detects that the link quality between the server and the PDA has increased.

2. This is reported to the Order Manager, who will map the new values to the quality level H (we can assume a sudden large connection improvement e.g., by entering the room where the server is placed).

3. Order Manager publishes the new quality level H in the Status Matrix.

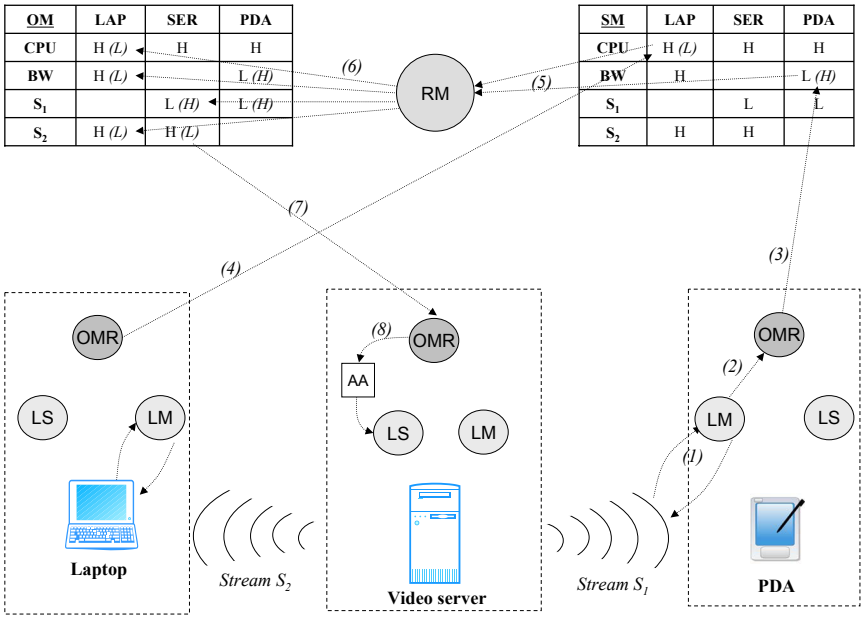

Figure 3.4: Example global adaptation

4. Assume also that there has been some change in the CPU availability on the laptop, i.e., it gets decreased from H to L due to some new, CPU intensive application that has started to run on the laptop. Initially, the local adaptation mechanism on the laptop will react to the changes in the CPU load by e.g., by performing selective frame skipping in the video decoder that is processing the stream S_2. However, at some point the CPU QoS threshold will be exceeded and the new QoS value will be calculated and published in the Status Matrix for the CPU.

5. The Resource Manager is notified about the new quality level values.

6. At this point, it is up to the Resource Manager to take a decision

about the resource reallocation. Considering the available bandwidth and the streams priorities, one solution could be to set the quality of S_1 to M (since it has lower priority), and leave the quality of S_2 unchanged. However, streaming the high quality video stream to the laptop may not be a good solution, since the CPU on the laptop is overloaded and video frames will be skipped anyway. Hence, the Resource Manager, who has the total resource usage view of the system, decides to set L for stream S_2. This decision will not only reflect the resource status on the laptop correctly, but it will also allow S_1 to be set to H (which can be done because the quality of the connection between the server and the PDA has been changed to H).

7. The Order Managers on respective devices are informed about the new values (arrows to the OMRs of the PDA and the laptop are omitted in the figure to ease the readability).

8. The Order Managers enforce the new settings via their local schedulers and application adapters. For example, in the case of the server, the stream application adapter will make sure to decrease the quality of stream S_2. This can be done in several ways, e.g., by reading a lower quality version of S_2 that has been stored on the server in advance, or by using an online modification of original S_2 by using the quality-aware preventive frame skipping methods that we have developed in our previous work [20].

3.7 Chapter summary

Applications executing in heterogenous, dynamic environments vary their resource demands over time while experiencing the uncertainty of execution environments. Yet, they must be capable to react to changes in the operating conditions and maintain required performance levels.

In this chapter we have developed a method for an efficient Quality-of-Service provision and adaptation in dynamic, heterogenous systems, based on our Matrix framework for resource management. It integrates local QoS mechanisms of the involved devices that deal mostly with short-term resource fluctuations, with a global adaptation mechanism that handles structural and long-term load variations on the system level.

The idea is to perform local adaptation as long as possible, using a control model for resource monitoring and adjustment, and if a resource availability passes the range of the currently assigned QoS level, the global adaptation mechanism takes over. We adopt a closed loop control model for local adaptation. So, available resources are continuously monitored by Local Monitors, and the changes are reported the Order Manager on the local device, which then determines if the resource fluctuation can be handled locally, by the Local Scheduler, or not.

The effectiveness of our proposed integrated QoS approach is illustrated in the context of video streaming.

Chapter 4

Evaluation

A comprehensive analysis, derivation of values and trade-off, require a complete implementation, which is still ongoing work. However, we have performed simulations in order to test the efficiency of the Matrix and the Integrated QoS Adaptation approach. For this purpose, we have implemented a simulator, described in this chapter, together with obtained simulation results.

4.1 Simulation setup

The simulator is a multi-threaded application, implemented in C. Resource Manager, as well as Order Managers are designed as separate threads. The Status and Order Matrix are data structures, while the Local Schedulers and Monitors are not implemented in this version of the simulator. In the simulator, each Order Manger represents one node in the system (e.g., every node has just one Order Manager). The amount of nodes in the system is variable and defined by an input parameter. Hence, by changing the value of the input parameter, we can change a number of Order Manager threads in the simulation. In this way, we have the possibility to simulate some aspects of our approach with different number of nodes involved. Moreover, we have implemented an inter process communication (IPC), that simulates some part of the publish/subscribe communication model. We have used mailslots in Windows, which is a mechanism for one-way interprocess communication. All simulations run have been performed on one PC computer with the processor speed of 1.7GHz.

4.1.1 Bandwidth estimation

In the simulator, information about available resources (in this case about available bandwidth in wireless network) is obtained from input data files. These files are results of another comprehensive simulation on bandwidth estimation performed in our lab. The applied bandwidth prediction method has used a packet-pair probing technique[1], combined with an exponential averaging method[2]. The equation for bandwidth prediction is following:

[1]The basic idea of a packet probing technique is to send two back-to-back prob packets to a neighbor node to measure their dispersion.

[2]Exponential averaging is technique to analyze and average a time series of data, while making taking into account both the previous and current values

$\mathcal{P}_k = \alpha \mathcal{BWT}_k + (1 - \alpha)\mathcal{P}_{k-1}$.

Where $\mathcal{P}_k$ and $\mathcal{P}_{k-1}$ stands for prediction, the current and previous one. α is a constant that indicates in which extent the history is important, and $\mathcal{BWT}_k$ is the current bandwidth measurement. For more details, please see [26, 40].

Basically, every file is a list of bandwidth estimations between two nodes connected through a 802.11g router (linksys wrt54g). Different input files correspond to different scenarios, ranging from a "free channel" to high cross traffic interferences. In this way we have been able to simulate fluctuation of available wireless network for different parts of network. For example, one of the files that we have used describes the available bandwidth according the Figure 4.1. The sampling interval is 120 milliseconds.

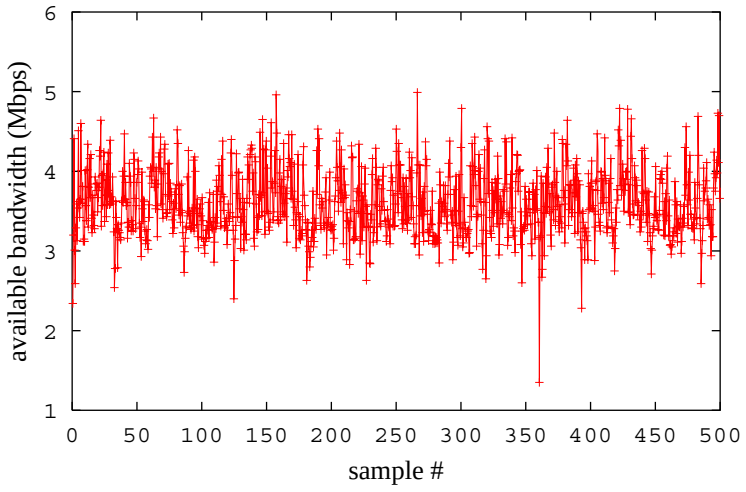

Figure 4.1: Available wireless bandwidth fluctuations

Accordingly, the functionality of Local Monitors (i.e. monitoring of available resources) is been simulated by using these input (log) files.

4.1.2 Video streams

We have used a MPEG-2 stream that has been extracted from a original DVD movie. More specifically, we have used information about one MPEG-2 stream's frames size, i.e., its resource demand, obtained during the performed analysis in [21]. Figure 4.2 shows variations in GOP size for one of the video streams that we have used in the simulations. A GOP (Group Of Pictures) in a MPEG-2 video consists of all the pictures (frames) that follow a GOP header before another GOP header. The GOP length is flexible, but quite common values are 12 and 15. In our case, one GOP consists of 12 pictures.

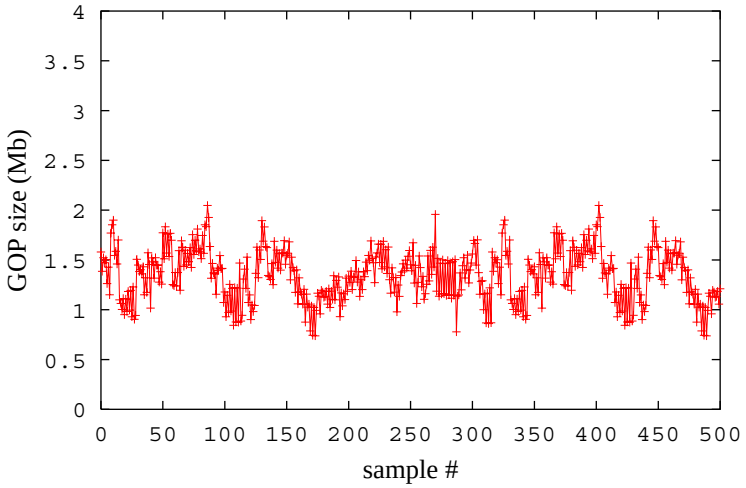

Figure 4.2: Stream demand (on GOP basis)

4.2 Simulations results

As we have mentioned before, we have evaluated our method in the context of video streaming. Here we present results from a 15 minutes video streaming simulation using our approach. For the sake of presentations

clarity, some of the figures show just one small part of the simulation results.

4.2.1 System state presentation

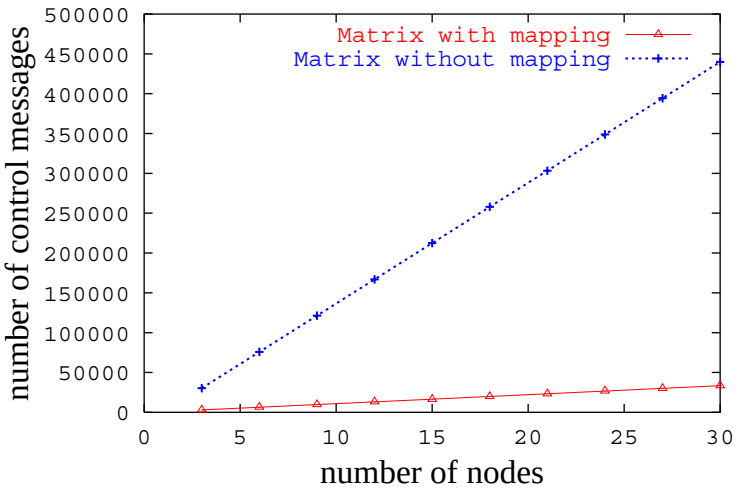

Figure 4.3: Reduced system state presentation

Figure 4.3 shows how the number of the control messages increase with the increased number of devices in the system, with and without mapping of available resources to quality levels. It is clear that we have achieved a significant reduction in the number of control messages by using the Matrix mapping approach.

Figure 4.4 depicts the result of simulation of the using the Matrix approach with different resource reallocation policies, i.e., naive, fair-prioritized, and greedy (explained in 3.4. The simulations are performed with two streams, and 3 quality levels. As we can see from the figure, a choice of resource reallocation policy does not influence too much the number of control messages.

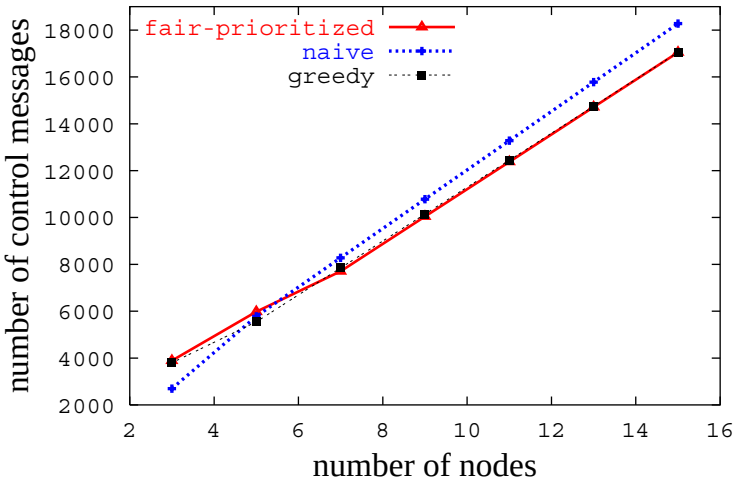

Figure 4.4: Resource reallocation policies and control messages

4.2.2 Spared resources due to the global system view

We have analyzed the difference between QoS levels determined on a device, and by the Resource Manager.

First, we have simulated the usage of 10 devices in the system and shown how an MPEG-2 video stream is adapted based on current resource availability (network bandwidth). The sampling interval is 480ms. The following quality levels for available bandwidth (given in Mbps) have been used:

$$\begin{aligned} q_1(BW) &= [1.5, 2.5] \ \ (L) \\ q_2(BW) &= [2.5, 4] \ \ (M) \\ q_3(BW) &= [4, 11] \ \ (H) \end{aligned}$$

Figure 4.5 shows QoS levels, which are decided on the device, based on its local view. Figures 4.6 and 4.7 present the QoS levels chosen con-

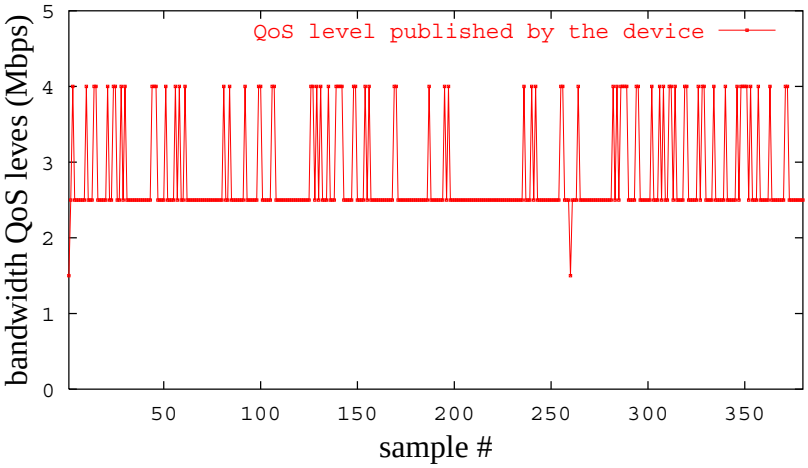

Figure 4.5: Local system view

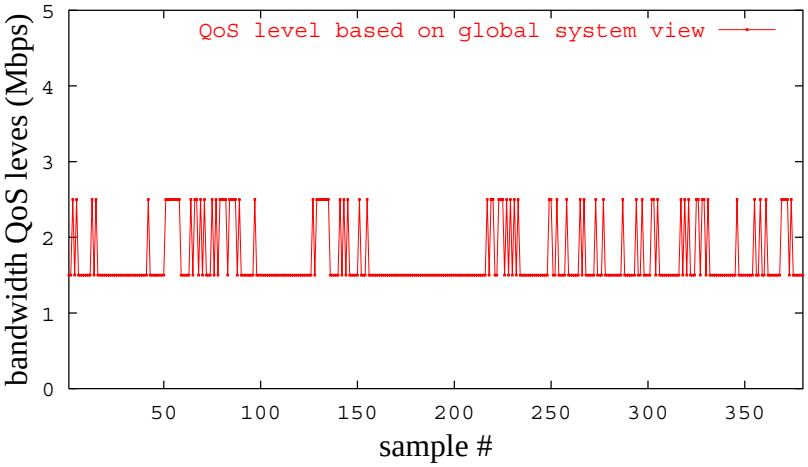

Figure 4.6: Global system view (one video stream)

currently by the Resource Manager for the device, after the performed global resource reallocation, with one respective two video streams in

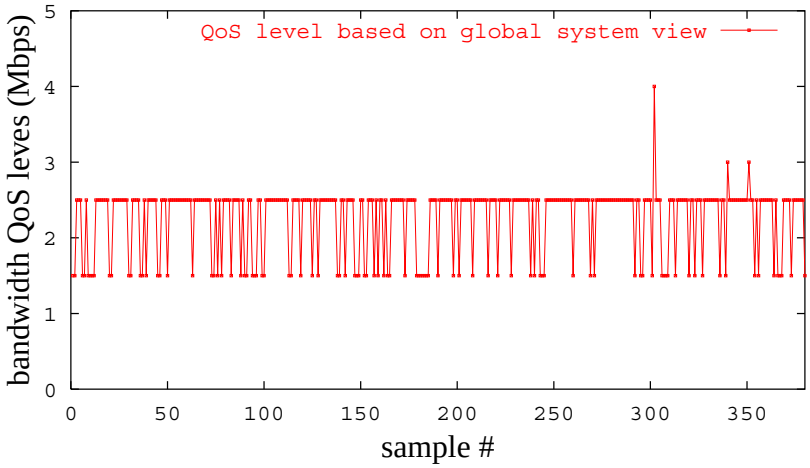

Figure 4.7: Global system view (two video streams)

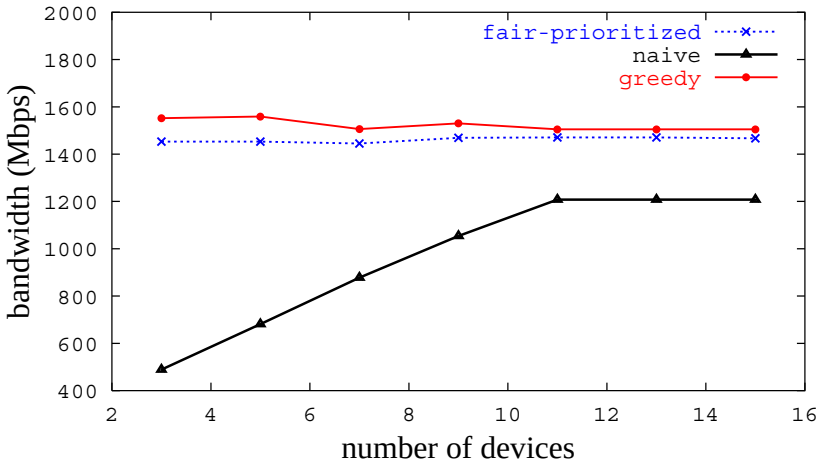

Figure 4.8: Spared resources (wireless bandwidth) on one device when applying the different resource reallocation policies

the system. The comparison between Figures 4.5 and 4.6, and also 4.5 and 4.7, give us an idea about the amount of possibly wasted resources on the device, due to the device's limited local system view.

In Figure 4.8, we present a total amount of the spared network bandwidth, on one device during the 15 minutes long simulation with two video streams in the system. We performed this simulation for different number of devices and we have applied three resource reallocation policies: naive, fair-prioritized, and greedy. The results point out that for the fair-prioritized and greedy policies a significant gain, in the spared resources, is obtained even with just a few devices in the system. While, for the naive method this gain is bigger when more devices are involved.

All presented simulation results (Figures 4.6, 4.7, 4.8), illustrate the efficiency of the Matrix approach where adjustment of resources is not only based on the limited local system view of one device (like in hop-by-hop approach), but also on the current available resources of all involved devices. In that way, our approach enables a system wide optimization.

4.2.3 Granularity issues

As discussed earlier, granularity issues are essential to provide an acceptably fresh view of the resources' states, both spatial and temporal. In the case of spatial granularity, on one side, if we use many quality levels, the freshness of the system state presentation will be good, but the overheads for state and order updates will overload the system. On the other side, having a too small number of quality levels (e.g., two) would imply significantly degradation of the freshness of the system view, but reduced overheads. Figures 4.9 and 4.10 show clearly, as expected, that number of the quality levels has an impact on the amount of control messages in the system. The larger number of quality levels, the increased amount of control messages in the system. The results from the performed simulation, both with one and two video streams, show the greatest deviation in the total number of control messages is between

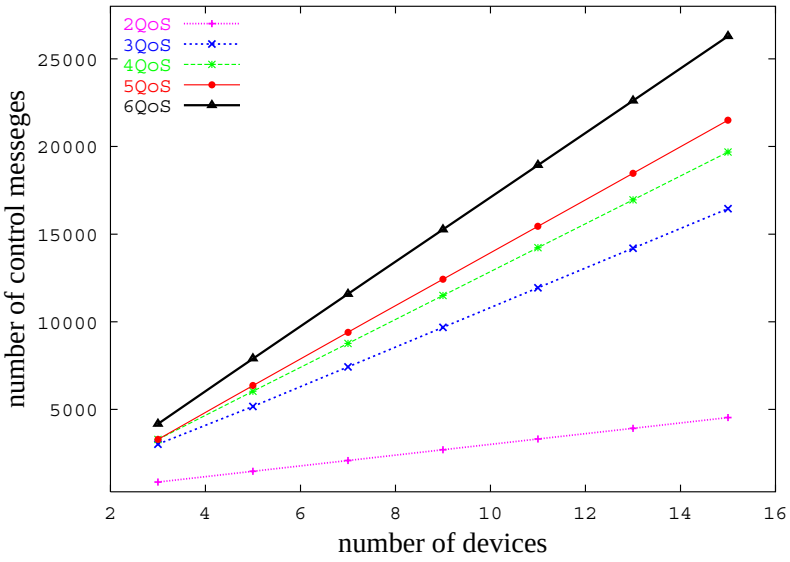

Figure 4.9: Control messages for different number of QoS levels (one video stream)

the lines representing two and three quality levels. While, this deviation is smaller among the lines representing the results for having three, four, five, or six quality levels. During the simulation we used the following thresholds for different QoS levels:

- two QoS levels
  $\{[1.5, 4.0), [4.0, 11.0)\}$
- three QoS levels
  $\{[1.5, 2.5), [2.5, 4.0), [4.0, 11.0)\}$
- four QoS levels
  $\{[1.5, 2.0), [2.0, 3.0), [3.0, 4.0), [4.0, 11.0)\}$
- five QoS levels

$\{[1.5, 2.0), [2.0, 2.5), [2.5, 3.0), [3.0, 4.0), [4.0, 11.0)\}$

- six QoS levels
  $\{[1.5, 2.0), [2.0, 2.5), [2.5, 3.0), [3.0, 3.5), [3.5, 4.0), [4.0, 11.0)\}$

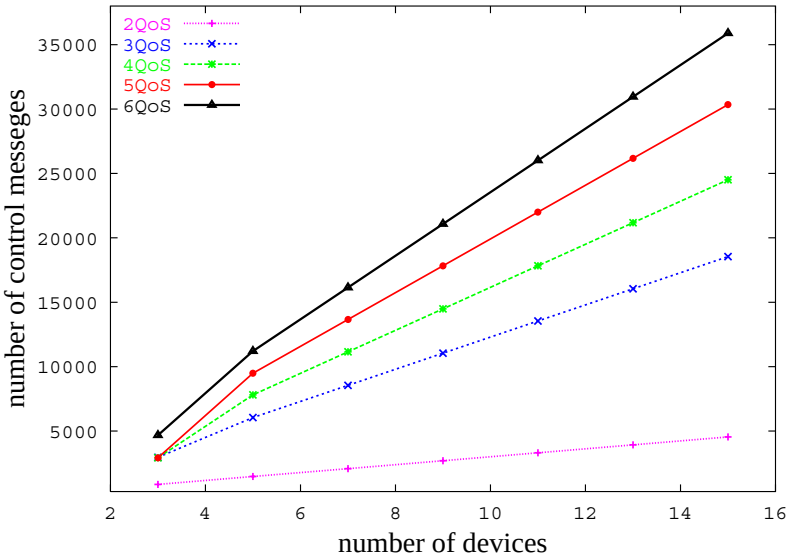

Figure 4.10: Control messages for different number of QoS levels (two video streams)

The obtained results are representative for our simulation setup. We are aware that the choice of settings for these thresholds for QoS levels, obviously has an impact on the results of the simulation, specially in the case when we have just a few quality levels (e.g., two QoS).

4.2.4 Global vs Local resource adaptation

As mentioned before, in the Integrated QoS Adaptation approach, we want to handle resource fluctuation as long as possible locally on the

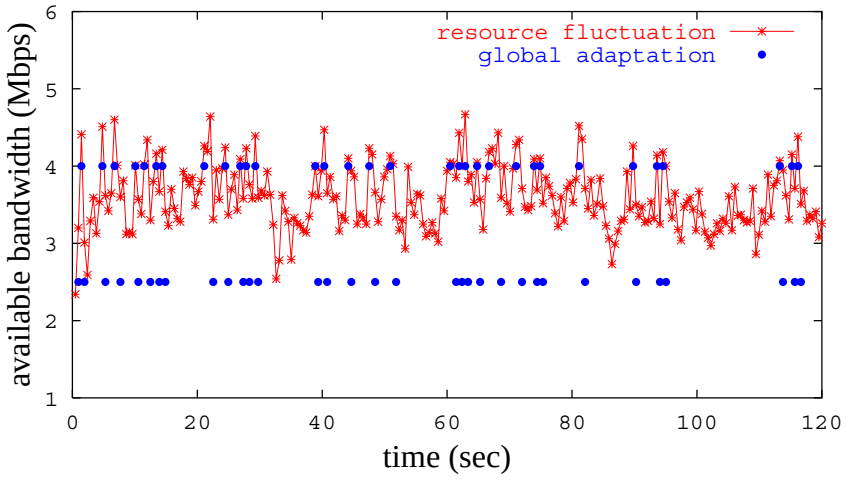

Figure 4.11: Invocation of global adaptation

devices. But, whenever a local resource availability exceeds the thresholds of the current QoS level, a global adaptation is involved.

Figure 4.11 shows at which frequency the local and global adaptation are involved during the simulations with one video stream. Although, it only depicts a small fraction of a 15 minutes long simulation, we can see in the same figure, the local adaptation mechanism is deployed most of the time, while the global mechanism is triggered only when necessary, i.e., the QoS has changed so much that the system reallocation must take place. During the 15 minutes simulation, 77 percent of the resource variations have been handled by the local adaptation, versus those 23 percent managed by the global adaptation.

4.3 Chapter summary

We have developed a simulator in order to evaluate the Matrix and the Integrated QoS Adaptation approach. The results from simulations show some clear advantages of the Matrix approach, such as reduced

system state presentations, reduced overheads, and system wide optimization.

Furthermore, we have looked at the cost, in terms of amount of control messages, that we have to pay for having different number of Quality-of-Service levels (QoS).

Moreover, we have evaluated how the different resource reallocation policies, that we have implemented in our Integrated QoS Adaptation approach, influence the amount of control messages in the Matrix framework. Also, we have shown at which frequency global vs local adaptation occurs in the Integrated QoS Adaptation approach. Clearly, resource adaptation takes place to the greatest extent on a device level (local adaptation).

The comparison between the Matrix and hop-by-hop approach has been analytically done in Chapter 2. In this chapter, we have not elaborated on that, since the current status of the simulator does not offer any possibility on further comparison between these two approaches.

In summary, the performed simulations have clearly stressed the notable characteristics of our approach, e.g., reduced overhead, system state presentation, or system wide presentation. Though, we realize that the simulator is an inadequate tool for performing any timing analysis, e.g., determination of resource reallocation delay. This type of evaluation requires a complete implementation of the Matrix framework.

Finally, we are aware that our simulation results depend on type of resource variations that we have in the system. That is to say, there is a strong connection between resource fluctuation and the number of state changes, as well as the corresponding resource management actions. However, the magnitude of the performed simulation, give us enough information to draw those above mentioned conclusions.

Chapter 5

Conclusions

The number of different applications that execute in open, dynamic and heterogeneous environments is constantly increasing. These applications usually vary in their resource demands over time, and, at the same time, they have to deal with the fluctuation of execution environments. An example of an open, heterogeneous, dynamic system is home networks. Home networks are characterized by highly fluctuating, and usually restricted system resources (e.g., wireless network). In addition, there exist a variety of devices connected together, which differ in many aspects, among other things in their performance capability. Despite all that, users are not willing to compromise on quality for multimedia (video) applications.

The work presented in this thesis has been initiated as part of a FABRIC project. One of the project's objectives was to provide high quality video streaming over heterogeneous networks, which requires efficient management of available resources.

We started by identifying crucial issues in order to achieve an efficient resource management: representation of the fluctuating system state, resource allocation decisions, and dissemination of orders. Then, we stressed the importance of finding tradeoffs between accuracy of system state information, and efforts to transport and process. Further, we proposed the Matrix, which is an adaptive framework for applying real-

time resource management methods for decoupled video streaming of heterogenous devices. In order to reduce the overhead for the system state presentation, we map the resource availability to just few QoS levels. In that way, we also filter out too high fluctuations that could overload the resource scheduling. The Matrix provides an interface to decouple device scheduling and system resource allocation. By having data handling at appropriate levels, i.e., resource management at global high level, while device specific information is processed locally at devices, in the Matrix we achieved a loose coupling between the system resource management and devices. Accordingly, the Matrix is based on a global abstraction of device states, which reduces system state information and decreases overheads for its determination and dissemination. We have also discussed architectural design aspects of the Matrix framework, and all its substantial parts.

Moreover, we studied the admission control and resource reallocation delay in the Matrix framework. We showed that both admission control and resource reallocation delay are entirely independent of the number of devices (nodes) present it the system, i.e., a larger number of devices in the system does not mean a large delay. This is a clear advantage of the Matrix compared to other approaches, in particular a hop-by-hop approach.

Theoretically, it is not necessary that all devices in the system have to use the Matrix. However, a mix of direct explicit communication and Matrix abstraction is conceivable for resource management, although benefits of the data abstraction would obviously be reduced. Thus, our advice is to have devices either fully attached to the Matrix, or not at all.

Apart from the resource management, the Quality-of-Service (QoS) adaptation is also a vital operation when maximizing the overall system quality. We have proposed a method for an efficient QoS provision and adaptation in dynamic, heterogeneous systems, which we call integrated QoS adaptation approach. It is based on the Matrix framework, which architecture enables handling of different types of load fluctuation (e.g., structural, stochastic) at different system levels. Consequently, we propose an integrated global and local QoS adaptation mechanism, where

the structural and long-term load variations are object for global adaptation, while the temporal load and short-term resource variations are taken care locally on devices. Basically, in our approach, we handle locally (at a device level) resource variations as long as these are within the range of a certain QoS level. Whenever a local mechanism detects that a local resource availability has exceeded the current QoS level, a global adaptation mechanism is involved. The task of the global adaptation is adjustment of the resource usage among all involved applications. In our approach, we let the user to decide if applications have priorities or not. However, we support user defined *priorities* between applications to be used when redistributing resources. Based on this, we present four reallocation policies in our approach, *naive, fair, fair-prioritized, and greedy*.

We validate our approaches by means of simulations experiments, in the context of video streaming. In addition, we have compared the Matrix to a hop-by-hop approach. Our simulation results clearly emphasized some advantages of the Matrix, such as reduced system state presentations, reduced overheads, and system wide optimization.

Furthermore, we looked into the relationship between QoS levels granularity and the freshness of the system state presentation. In particular, we look at how the increased number of QoS levels influences the amount of control messages in the system. Also, we have evaluated the different resource reallocation policies, and looked at what frequency global vs local adaptation occurs in the Integrated QoS Adaptation approach. Simulations results underline that resource adaptation takes place mostly on a device level (local adaptation).

In this thesis, we chose home networks and video streaming as the application domain for our proposed approaches. However, we do not se any limitation to expand the usage of our approach to the health sector, or some other community social/industrial applications. Resource management and QoS adaptation are a must whenever we are surrounded with heterogenous, mobile, and dynamic environments.

Therefore, one task of our future work will be exploiting the proposed framework in other application domains than in home networks.

Moreover, we will look into possibilities of further developing the local control model by formally describing the system's behavior with a set of differential equations. Also, we are working on a more general model for mapping between resource demands and abstract QoS levels. Finally, we are working on a full implementation of the Matrix framework.

Appendix A

Implementation Details

The Matrix framework is quite complex and we are still working on its full implementation. However, the hierarchical architecture and the loose coupling between system modules makes it possible to work on different parts independently of each other. In this chapter we will give an overview of the implemented parts, and also we are going to present and discuss two different publish/subscribe mechanisms and their application with the Matrix framework.

A.1 Implemented Modules

Current implementation includes Resource Manager, Local Monitors and Local Schedulers for CPU and network bandwidth, and an Application Adapter (Video Stream Adapter).

A.1.1 Resource Manager

Resource Manager is a crucial part of the Matrix framework. Still, its implementation is quite straightforward. Basically, it consist of those four resource reallocation policies (i.e., naive, fair, fair-prioritized, and greedy) that we support, and a communication specific client code, in order to adopt a certain publish/subscribe mechanism.

A.1.2 Local Network Scheduler

For network scheduling we use the traffic shaping approach, which provides different QoS by dynamically adapting the transmission rate of nodes, to match the currently available bandwidth of a wireless network. The Traffic Shaper adjusts the outbound traffic accordingly to input parameters (i.e., the amount of available bandwidth assign to the Local Scheduler). In this architecture priority is given to a real-time traffic over non real-time traffic. Please see [26] for full implementation details of the Traffic Shaper.

A.1.3 Local Network Monitor

For monitoring and estimation of available bandwidth (over 802.11b wireless Ethernet), we use a method that provides us with the average bandwidth that will be available during a certain time interval. The architecture consists of a bandwidth predictor that first uses a simple probe-packet technique to predict the available bandwidth. Then, exponential averaging is used to predict the future available bandwidth based on the current measurement and the history of previous predictions. Also, we refer to [26] for details.

A.1.4 Local CPU Scheduler

The allocation of CPU to the applications depends on the scheduling mechanism that is used. We have developed a predictable and flexible real-time scheduling method that we refer to as *slot shifting* [19]. The basic idea is to guarantee a certain quality of service to applications before run-time, and then adjust it at run-time according to the current status of the system. The full details about the scheduler and its application in the context of media processing can be found in [21].

Furthermore, we are even looking into other scheduling methods. One suitable technique that fits our needs *elastic scheduling approach* [9]. It handles overload through a variation of periods and in that way manage to decrease the processor utilization up to a desired level.

A.1.5 Local CPU Monitor

Since we use a real-time scheduling mechanism, the CPU monitoring is very simple to achieve. The *spare capacity* mechanism of slot shifting provides easy access of the amount and the distribution of available resources at run-time [19].

A.1.6 Video Stream Adapter

We have implemented an Application Adapter for MPEG-2 video stream adaptation, based on quality-aware, selective frame skipping. Order Manager sends allowed abstract quality level to the video adapter, which then adjusts the stream according to available resources by skipping the least important video frames, see Figure A.1. For the frame priority assignment algorithm we have proposed a number of criteria to be applied when setting priorities to the frames. Please see [20] for details on skipping criteria and full analysis.

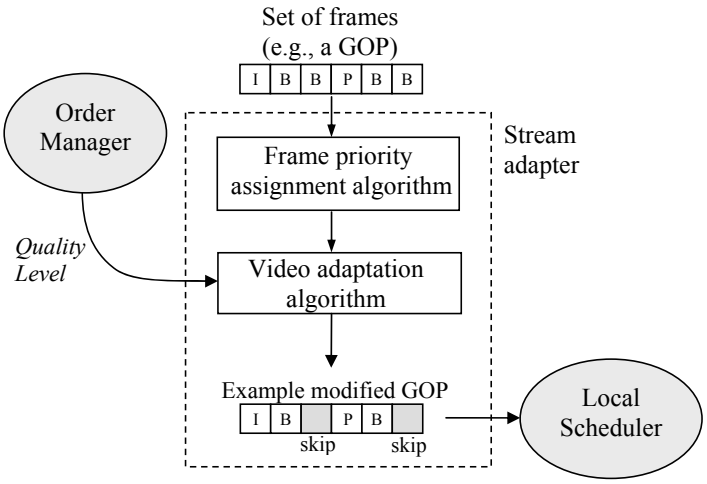

Figure A.1: MPEG-2 video adaptation in Matrix

A.1.7 The Matrix data structures

AppInfo - This record contains information about a application.

- *appID* - Each application ID is allotted by the order managers on the device that is s source device for the application. If the device can produce one application (stream) with different qualities, each version of application will get a unique ID.
- *appQoS* - The application's resource requirements, expressed in the QoS levels.
- *appPriority* - A user defined priority for the application.

MatrixCell - This class is base class for StatusMatrixCell and OrderMatrixCell classes. The following attributes are provided:

- *OrderManagerID* - The unique identifier of an order manager

- *application* - An AppInfo record, for a application that is relevant for this Order Manager.

- typeOfDevice - Describes is a source, sink, or just device on play-out route.

StatusMatrixCell - This class represents an entry in the Status Matrix. This class is subclass of the MatrixCell class and inherits all its attributes. Each resource is represented by the following attributes:

- *Current value* - Current resource availability. This value is out of the limited number range (QoS performance level).

- *Current granularity* - It is the time interval until which the current value is likely to not change.

- *Likelihood* - This attributes gives the probability that the given value under attribute "current granularity" will hold.

OrderMatrixCell - This class represents an entry in the Order Matrix. This class is derived from a MatrixCell class. Each this entry is one order from the resource manager for resource allocation. The attributes of this class are:

- *Delay* - It is a sub delay for one device in the stream path

- *Value* - This value is out of the limited number range (QoS performance level).

Status and Order Matrix - Collections of StatusMatrixCell/OrderMatrixCell, which could be implemented in different ways, for example as an array or list data structure.

A.2 Publish/Subscribe Mechanisms

In section 2.4 we have discussed some of advantages of publish/subscribe communication mechanism in the context of dynamic mobile environment (e.g., home networks). We have also mentioned that at present we are using these publish/subscribes models only for signaling, i.e., management part of the Matrix framework, not for a transport of streams.

In this section, we will give a brief description of two publish/subscribe mechanisms that we have look into so far. The first one is High Level Architecture (HLA), a commercial middleware standard, and second one is an open source MOM (Message Oriented Middleware), called Xml-Blaster.

A.2.1 The Matrix and HLA

During the FABRIC project we have implemented a mock-up of Matrix approach, by using HLA (High Level Architecture), which is an interoperability standard, based on an anonymous publish/subscribe mechanism [1]. Its publish-subscribe model connects anonymous information producers (publishers) with information consumers (subscribers). In the terminology of HLA, individual simulations are known as federates. The collection of federates brought together to simulate a complex environment is known as a federation.

In the Matrix's mockup [39], Resource Manager and Order Mangers are implemented as federates. The Status Matrix contains information produced by various nodes in the system. Therefore the Status Matrix can not be a federate. Rather, we present the Status Matrix as a collection of objects. Thus, the elements of Status Matrix are published by the Order Managers placed on nodes. Each Order Manager adds an entry in the Status Matrix and becomes owner of this object. The Resource Manager, is interested in information published in the Status Matrix, i.e. it subscribes to the Status Matrix.

The Order Matrix is represented as a collection of objects too. The elements of the Order Matrix will be published by the Resource Manager.

As discussed in 2.4, HLA's publish/subsrcibe communication model, as decoupling of devices, fits well with the Matrix. One additional good point of HLA for the Matrix concept is the dynamic addition and removal of members of a federation. This HLA's characteristic is well suitable for a redundancy of devices in the Matrix. Federates (order managers or a resource manager on a device) can fail without a major impact on the overall functionality. The lost federate (device) can be replaced during run time with some shadow federate (device).

A.2.2 The Matrix and XmlBlaster

As mentioned before, XmlBlaster is MoM (Message oriented Middleware), which also supports a publish/subscribe communication model. Publisher and subscriber are XmlBlaster clients that exchange messages via a XmlBlaster server. XmlBlaster server is a 100% Java MoM server, while clients can be written in many different languages. Moreover, publisher and subscribers communicate with the server independently, without having knowledge about each others. In that way, they make use of one of the main advantages of the publish/subscribe model, i.e., device decoupling.

XmlBlaster and its publish/subscribe communication model fits also well with the Matrix framework. Consequently, the Resource Managers and Order Managers are designed as XmlBlaster clients. Communication between Order Managers and the Resource Manager goes through XmlBlaster server. The Status and Order Matrix are collection of data produced by different nodes (Order Managers), and in this version of the Matrix implementation, they are implemented as arrays (data structure). The Local Monitors and Local Schedulers as threads. Hence, exchange of information between the Resource Manager and the Order Managers is handled by publish/subcribe mechanism, while communication between the Order Managers, Local Monitors, and Local Schedulers can be done with some form of Inter Process Communication (IPC), e.g., sockets, mailslots, pipes (see Figure A.2).

All good points mentioned for HLA are also valid in the case of

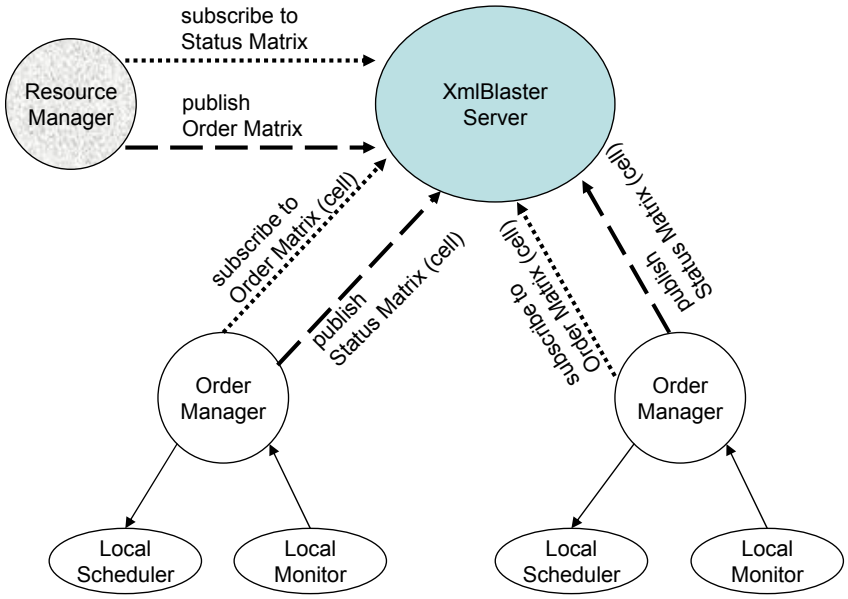

Figure A.2: Matrix and XmlBlaster

XmlBlaster. Moreover, XmlBlaster have some advantages compared to HLA. First, it is quite simple to use, and it gives support for many different programming languages and different protocols. Then, in addition, XmlBlaster is a open source middleware, while HLA is originally designed to be used by US Department of Defense.

Bibliography

[1] IEEE standard for Modeling and Simulation, High Level Architecture (HLA) - federate interface specification, no.:1516.1-2000.

[2] xmlBlaster - Message Oriented Middleware (MOM).

[3] C. Aurrecoechea A. Campbell and L. Hauw. A survey of QoS architectures. In *Multemedia Systems*, volume 6, pages 138–151, 1998.

[4] Siemens AG. SURPASS Home Entertainment White Paper. April 2005. `http://networks.siemens.com/voip/carrier-en/products-solutions/surpass-home-entertainment/surpass-home-entertainment.html`.

[5] Gregor v. Bochmann Jan Gecsei Andreas Vogel, Brigitte Kerhervé. Distributed multimedia applications and quality of service: a survey. In *Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada*, page 71, October 1994.

[6] Geoff Coulson Andrew Campbell and David Hutchison. A quality of service architecture. In *ACM SIGCOMM Computer Communication Review*, volume 24, pages 6–27, 1994.

[7] Gillian M. Wilson Anna Bouch and M. Angela Sasse. A 3-dimensional Approach to Assessing End-User quality of Service.

In *Proceedings of the London Communications Symposium*, pages 47–50, Sept 2001.

[8] Reinder J. Brill. *Real-time scheduling for media processing using conditionally guaranteed budgets*. Ph.D Thesis, Technical University Eindhoven, 2004.

[9] G.C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. In *IEEE Transactions on Computers*, volume 51, pages 191–302, March 2002.

[10] Lu Xichen Chen Xiaomei and Wang Huaimin. The design of QoS management framework based on CORBA A/V STREAM architecture. In *High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. The Fourth International Conference/Exhibition on*, May 2000.

[11] H. Chu, Klara Nahrstedt, and Srinivas Narayan. QoS-aware resource management for distributed multimedia applications, January 1998.

[12] Gianpaolo Cugola and H.-Arno Jacobsen. Using publish/subscribe middleware for mobile systems. In *SIGMOBILE Mob. Comput. Commun. Rev.*, volume 6, pages 25–33, New York, NY, USA, 2002. ACM.

[13] R. Wahbe R. Govindan D. P. Anderson, S. Tzou and M. Andrews. Support for Continuous Media in the Dash System. In *In ICDCS10*, pages 54–61, May 1990.

[14] Murata M. Fukuda K., Wakamiya N. and Miyahara H. QoS Mapping between User's Preference and Bandwidth Control for Video Transport. In *Proc. 5 th International Workshop on Quality of Service (IWQOS'97), Columbia University, New York, US*, 1997.

[15] G. Gopalakrishna and G. Parulkar. Efficient Quality of Service in Multimedia Computer Operating Systems. Technical report,

Department of computer science, Washington University, August 1994.

[16] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe in a mobile environment. In *Wireless Networks*, volume 10, pages 643–652. Springer Netherlands, 2004.

[17] Jean-Francois Huard and Aurel A. Lazar. On QOS Mapping in Multimedia Networks. In *Proceedings of the 21st International Computer Software and Applications Conference*, page 312, 1997.

[18] B. Deffner I. Busse and H. Schulzrinne. Dynamic qos control of multimedia applications based on rtp. In *Computer Communications, 19(1):49-58, Jan. 1996.*

[19] D. Isovic and G. Fohler. Efficient scheduling of sporadic, aperiodic, and periodic tasks with complex constraints. In *21st IEEE RTSS*, USA, November 2000.

[20] D. Isovic and G. Fohler. Quality aware MPEG-2 stream adaptation in resource constrained systems. In *ECRTS*, Catania, Italy, July 2004.

[21] Damir Isovic. *Flexible Scheduling for Media Processing in Resource Constrained Real-Time Systems*. PhD thesis, 2004.

[22] ITU-Rec. BT.500-8, Methodology for the subjective assessment of the quality of television pictures, Geneva, Switzerland, 1998.

[23] ITU-T. Recommendation E.800, International Telecommunication Union, Geneva, Switzerland, 1994.

[24] ITU-T. Recommendation P.910,International Telecommunication Union, Geneva, Switzerland, 1996.

[25] Andreas Kassler, Andreas Schorr, Christoph Niedermeier, Reiner Schmid, and Andreas Schrader. MASA - A scalable QoS Framework. In *Proceedings of Internet and Multimedia Systems and Applications (IMSA), Honolulu, USA*, August 2003.

[26] Tomas Lennvall and Gerhard Fohler. Providing adaptive qos in wireless networks by traffic shaping. In *Resource management for media processing in networked embedded systems (RM4NES)*, Eindhoven, Netherlands, March 2005.

[27] B. Li and K. Nahrstedt. A control-based middleware framework for quality-of-service adaptations. In *Selected Areas in Communications, IEEE Journal*, volume 17, pages 1632–1650.

[28] B. Li and K. Nahrstedt. Impact of control theory on qos adaptation in distributed middleware systems. In *American Control Conference, 2001. Proceedings of the 2001*, 2001.

[29] Wilfried Osberger Mark Masry, Sheila S. Hemami and Ann Marie Rohaly. Subjective Quality Evaluation of Low Bit Rate Video. In *SPIE Conf. on Human Vision and Electronic Imaging, vol. 4299, San Jose, CA*, volume vol .4299, 2001.

[30] D. Miras, B. Teitelbaum, A. Sadagic, J. Leigh, M.El Zarki, and H. Liu. A Survey on Network QoS needs of Advanced Internet Applications. Working Document of Internet 2. Dec 2002.

[31] Daniel Mosse and Ha Ly. User to Network QoS Parameter Transformation in Networked Multimedia Systems. In *ON-LINE PROCEEDINGS, IEEE Real-Time Systems Symposium, Workshop on Resource Allocation Problems in Multimedia Systems, Washington*, December 1996.

[32] Klara Nahrstedt and Jonathan M. Smith. Design, Implementation an Experiences of the OMEGA End-Point Architecture. Technical report, Distributed Systems Laboratory, University of Pennsylvania, Philadelphia.

[33] Nortel Networks. Introduction to Quality of Service (QoS). Technical report. `http://www.nortel.com/products/02/bstk/switches/bps/collateral/56058.25_022403.pdf`.

[34] Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, and Kevin R. Walker. Agile application-aware adaptation for mobility. In *16th ACM Symposium on Operating Systems Principles*, France, 1997.

[35] Pascal Frossard Olivier Verscheure and Maher Hamdi. User-oriented QoS Analysis in MPEG-2 Video Delivery. 1999.

[36] Clara Otero Perez, Liesbeth Steffens, Peter van der Stok, Sjir van Loo, Alejandro Alonso, José F. Ruíz, Reinder J. Bril, and Marisol García Valls. QoS-based resource management for ambient intelligence. In *Ambient intelligence: impact on embedded system design*. Kluwer Academic Publishers Norwell, MA, USA, 2003.

[37] IST project OZONE. `http://www.extra.research.philips.com/euprojects/ozone/`.

[38] Philips Research. `http://www.research.philips.com/technologies/syst_softw/ami/vision.html`.

[39] L. Rizvanovic and G. Fohler. The MATRIX: A qos framework for streaming in heterogeneous systems. In *International Workshop on Real-Time for Multimedia,in conjunction with ECRTS04, Catania, Italy*, 2004.

[40] Carmen Ederra Sáez. Testbed for wireless available bandwidth estimation system, bachelor thesis. http://www.idt.mdh.se/utbildning/exjobb/files/TR0463.pdf.

[41] M. Shankar, M. De Miguel, and J.W.S. Liu. An end-to-end QoS management architecture. In *Real-Time Technology and Applications Symposium 1999*, 1999.

[42] J.A Stankovic, T. Abdelzaher, M. Marleya, G. Tao G, and S. Son. Feedback control scheduling in distributed real-time systems. In *RTSS*, December 2001.

[43] C. J. van den Branden Lambrecht and O. Verscheure. Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System. In *Proceedings of the SPIE, San Jose, CA*, volume vol .4299, pages 450–461, Feb 1996.

[44] Anna Watson and M.Angela Sasse. Measuring Perceived Quality of Speech and Video in Multimedia Conferencing. 1998.

[45] T. Yamazaki and J. Matsuda. On QoS Mapping in Adaptive QoS Management for Distributed Multimedia Applications. In *Proc. ITC-CSCC'99, vol.2*, pages 1342–1345, 1999.