Reprint: 33rd Annual IEEE Hawaii International Conference on System Sciences, HICSS, 2000

Quality of Service in Mobile and Wireless Networks: The Need for Proactive and Adaptive Applications

Marc Bechler, Hartmut Ritter, Jochen H. Schiller
Institute of Telematics, University of Karlsruhe
[mbechler | ritter | schiller]@telematik.informatik.uni-karlsruhe.de

Abstract

Although there are already many fixed networked computers around the world, we will see several orders of magnitude more wireless and mobile devices connected to the Internet. Each car, truck, aircraft, but also many people will carry a plethora of different devices for communication. However, all mobile and wireless systems have in common that the quality of their connectivity may change rapidly over time. Still, users want to get the best quality possible or the quality they pay for respectively. In this paper we describe the implementation of adaptive and proactive applications that are supported by our architecture. The user can express his or her preference by simply clicking on a Q(uality)-Button; the process of adaptation is performed without any user interaction. Feedback loops connecting applications, operating system, and network system realize the adaptation. We show how our architecture supports traditional adaptive applications and demonstrate the benefits of proactive applications in wireless and mobile environments.

1. Introduction

History shows that there is not one single networking technology covering all aspects of communication, the future will bring even more technologies. Examples are broadband infrastructures based on ATM or IP over Sonet, LANs based on Ethernet, wireless LANs according to IEEE 802.11, infrared access using IrDA [1], mobile and cordless phone systems (GSM, D-AMPS, DECT, PHS etc.). New technologies include Personal Area Networks using, e.g., Bluetooth [2], scalable QoS provisioning in the Internet with Differentiated Services [3, 4], and advanced protocol stacks for wireless and mobile communication (e.g., Wireless Application Protocol, WAP [5]). A single technology can never cover all aspects of communication. While in wired communication we could think of an ideal world with fiber to the desktop and powerful communication systems, the wireless and mobile domain puts several limitations on communication systems. Portability of devices restricts the performance, mobility imposes new requirements on QoS support, and wireless access limits the bandwidth (particularly at a high relative speed).

Particularly, in mobile and wireless environments several network technologies coexist, such as IrDA (Infrared Data Association), DECT (Digital Enhanced Cordless Telecommunications [6]), GSM (Global System for Mobile communications [6]), DAB (Digital Audio Broadcasting [6]), or wireless LANs (e.g., WaveLAN [7]). Depending on the current location, the user or mobile device can choose one (or more) of the available transmission links for communication with other mobile devices or for getting connected to the Internet.

Although wireless networks and mobile communications can be used for many applications, particular application environments seem to be predestined for their use – one of these environments is road traffic. Tomorrow's cars will comprise many wireless communication systems and mobility aware applications. Music, news, road conditions, weather reports, and other broadcasted information is received via DAB with 1.5 Mbit/s. For personal communication, a GSM phone might be available offering voice and data connectivity up to 115 kbit/s. For remote areas, satellite communication can be used, while the current position of the car is determined via GPS (Global Positioning System). Additionally, cars driving in the same area build a local ad-hoc network for fast information exchange in emergency situations or to help each other keeping a safety distance. In case of an accident, not only the airbag will be triggered, but also an emergency call to a service provider informing ambulance and police. Cars with this technology are already available. Future cars will also inform other cars about accidents via the ad-hoc network to help them slowing down in time, even before a driver can recognize the accident. Nowadays, busses, trucks, and trains are already transmitting maintenance and logistic information to their home base, which helps to improve organization (fleet management), and thus saves time and money.

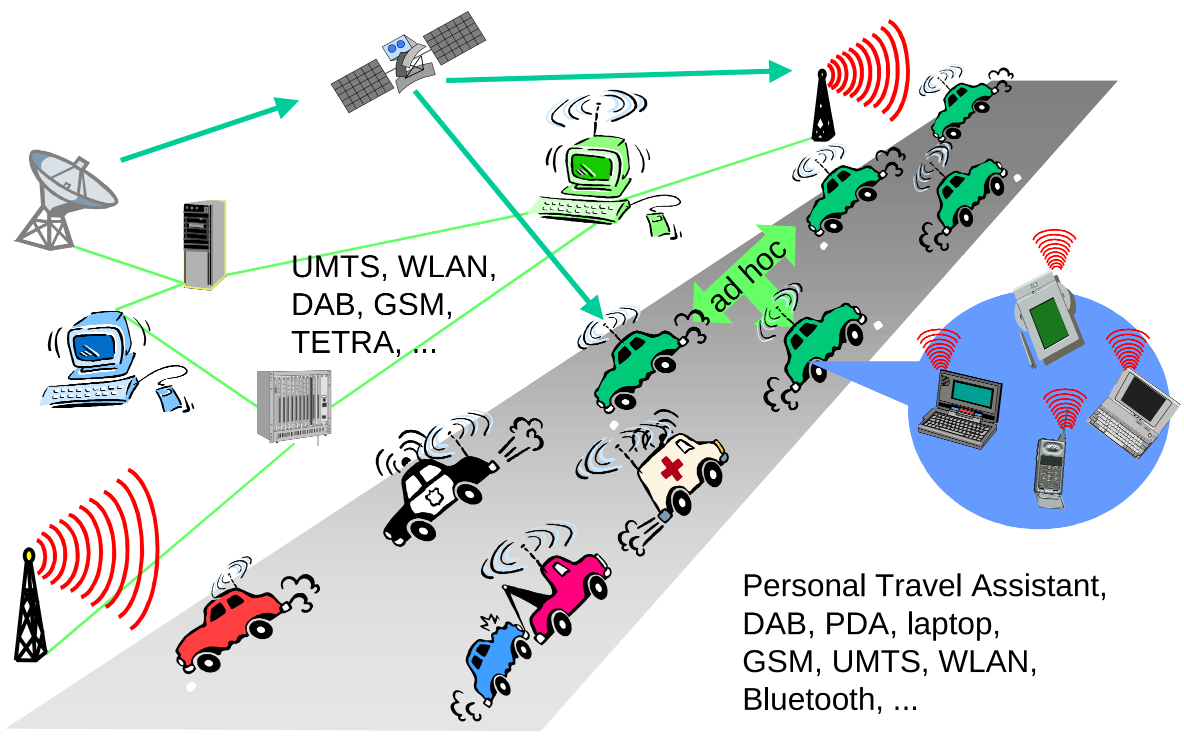

Figure 1. A typical application of mobile communications: road traffic

Figure 1 shows a typical scenario for mobile communications with many wireless devices which we currently investigate together with several industry partners. Networks with a fixed infrastructure like cellular phones (GSM, UMTS: Universal Mobile Telecommunications System [6]) will be interconnected with trunked radio systems (TETRA: Terrestrial Trunked Radio [6]) and wireless LANs (WLAN). Additionally, satellite communication links can be used. The networks between cars and also inside a car will more likely work in an ad-hoc fashion. Wireless pico networks inside a car can comprise PDAs, handhelds, Laptops, or mobile phones, e.g., connected with each other using the Bluetooth technology. Users may enter their car in the morning, the DAB receiver has already cached the current road and weather conditions, while a PDA may upload new destinations to the car's onboard navigation system. A Personal Travel Assistant (PTA) may support a user planning his or her trip. Traffic congestion on the highways will be signaled via DAB to the car, the car forwards selected information to the PTA which calculates a new route, and to the PDA which automatically rearranges the schedule for meetings. The new schedule is uploaded to the company's database using GSM.

Think of similar scenarios for air traffic or railroad traffic. However, different problems occur here due to the speed. While aircraft typically go up to 900 km/h, current trains up to 350 km/h, many technologies cannot operate if the relative speed of a mobile device exceeds, e.g., 250 km/h for GSM, or 100 km/h for AMPS. Only some technologies, like the Digital Audio Broadcasting (DAB) work up to 900 km/h.

This scenario shows that on the one hand, there is not one single technology that covers all aspects of communication. Each technology has its specific advantages and disadvantages regarding the communication QoS (Quality of Service). In contrast to wired network technologies, the QoS of existing wireless communication systems is rather limited and varies over time, e.g., low bandwidth, no guarantees for delay or jitter.

On the other hand, multimedia applications need a certain amount of computation time to meet their (application-specific) QoS parameters. Due to varying transmission characteristics in wireless environments (ranging from short-term degradation to the complete disconnection), the support of QoS in the operating system of the mobile end system plays an essential role to meet these requirements of users or applications.

This paper is organized as follows. After the introduction, the second chapter discusses the requirements for mobile devices and the benefits and differences of adaptive and proactive applications in comparison to common applications. Chapter 3 presents our solution for a simple, intuitive, and easy-to-use interface to express QoS demands – the Q-Button. The interface and the feedback loops controlling system resources are implemented and tested within our UNIQuE testbed are also described in this chapter. Finally, first results show the benefit of our approach and indicate future directions for research and performance evaluation.

2. Requirements for mobile devices

In contrast to fixed and wired workstations, the resources of handheld devices or palmtops, e.g., CPU performance, available memory, or network subsystem, are rather limited due to the restricted power supply. Thus, applications on portable devices generally provide a lower level for quality of service. Compared to wired workstations, e.g., the frame-rate of a video player running on a palmtop is by far lower as there is not enough CPU power to compute the frames within the required time-bound. Another aspect is that the QoS of a wireless link can extremely vary over time. Using adaptive applications, this QoS degradation can be compensated for a short-term period without a noticeable degradation of the application-specific QoS. As an example, a video player can briefly reduce the color depth, which is hardly recognized by the user. However, long-term distortions result in a significant degradation of the application-specific QoS, which manifests in, e.g., a reduced frame-rate. In order to prevent such a long-term degradation, it is very important that an application is able to control the sharing of local resources to meet its own QoS requirements. In the following, this kind of application, that actively takes effect on the sharing of a resource in advance, is called proactive application. This is in contrast to adaptive applications that try to adapt to the current situation but cannot actively control resources.

Current multitasking operating systems running on mobile devices (such as Windows CE) do not offer well-suited mechanisms to applications for achieving the requested support for quality of service. For wired workstations, several QoS extensions and QoS architectures (e.g., AQUA [8]), or completely new QoS-based operating systems (e.g., Nemesis [9]) exist that allow to specify many QoS parameters. For the use in mobile end-systems, those approaches are not suitable as they require too much memory and are too complex for the limited CPU power. For mobile devices, we need an proactive system comprising the following components:

- A simple, efficient, and memory-saving mechanism is needed that allows an application to control the sharing of the resources. Therefore, we propose an extensible architecture based on layered feedback mechanisms to achieve such an adaptive system as described in the next section.
- Due to the lower level of application-specific QoS, user requirements play a very important role and must be taken into account. In our architecture the user has to be able to specify her or his current preferences. S/he might selectively prefer some running applications, e.g., the current telephone call, while other applications (often best effort applications such as e-mail) are not so important and should get less resources. However, several applications may require an adaptive sharing of computation time to handle the sent and received data.
- The user interface has to be intuitive and easy-to-use. Especially for mobile devices used in, e.g., cars, a minimum of user interaction is essential as the user has to concentrate on the traffic.

2.1. Adaptive and Proactive Applications

The network characteristics in wireless environments are exposed to heavy variations ranging from slight degradations of the quality to a complete loss of the communication link. It is up to the end-system to compensate for these variations to provide an acceptable quality of service to the user. One main component that allows an improved QoS support is the application itself. In order to evaluate the suitability of applications running on mobile devices, we distinguish three types of applications: common applications, adaptive applications and proactive applications.

A common application does not perform any actions to adapt itself to the current availability of resources. If a resource becomes scarce, the quality of the application that uses this resource will be degraded. A typical example for this kind of application is a telephony application that does not perform any compression and uses Voice over IP (VoIP) for data transmission. For the duration of a telephone call a communication link with a capacity of 64 kbit/s and minimal jitter is needed to achieve acceptable results that satisfy the user (depending on the coding scheme). However, in mobile environments external disruptions usually affect the quality of the communication link that possibly leads to high jitter of the delivered packets or even to packet loss. Thus, the audio stream that reaches the recipient becomes more and more incomprehensible. The rather inflexible behavior of common applications makes them not well-suited for mobile communication scenarios where transmission quality may change rapidly over time.

Compared to common applications, the behavior of adaptive applications is more intelligent. These applications are characterized by the ability to react to changes in the quantity of resources. Adaptive applications support the user with the best service that is possible under the current circumstances. This behavior is advantageous in mobile environments with varying network characteristics. Think of a user roaming with his/her mobile device in overlaid and thus heterogeneous wireless networks. The mobile device could be connected to a high speed communication link inside a building, such as WaveLAN with a data rate of 2 Mbit/s. If the user leaves the WaveLAN cell, the communication link will be handed over to a narrow band communication link, e.g., GSM using 13.2 kbit/s for telephony services and 9.6 kbit/s for data services. In this case, an adaptive application adapts itself to this new situation caused by the vertical handover. In our telephone example it is conceivable that the application performs the compression algorithms used for encoding audio streams in GSM networks. Note that adaptation is a passive and thus reactive process. The application tries to support the best quality of service that is possible by utilizing the available resources. Thus, adaptation can deliver an improved QoS support to the user. Changes in the environment do not necessarily cause a degradation of the application QoS support. Even if a degradation of QoS is unavoidable, e.g., as a result of too limited resources as shown in the example, the level of QoS support is also improved compared to common applications. While an uncompressed audio stream over a 13.2 kbit/s communication link is nearly incomprehensible, a telephone call via GSM supports an acceptable quality of the audio data. Additionally, an adaptive application can decide to terminate itself if the monitored application-specific parameters that represent the quality of service fall below a predetermined threshold. More detailed information for describing the process of adaptation to varying network characteristics with the help of mathematical models can be found in [10]. In this work several mathematical models are examined which achieve a stable but agile behavior of the adaptation process.

Unlike adaptive applications, proactive applications actively affect the scheduling of a resource. Especially for mobile devices, this is a great advantage in multitasking environments such as the Windows CE operating system, where several applications compete for the limited resources, e.g., bandwidth and processing time. If resources become scarce, a proactive application first tries to adapt to the new situation, as described for the adaptive applications. If this adaptation is not enough to meet the QoS requirements specified by the user or the application, the proactive application influences the scheduling of resources in a second step to achieve a better QoS support for the user. This enables an application to meet its application-specific QoS parameters for a long-term period, even if more applications compete for the available resources. Note that this kind of intervention does not improve the availability of resources. Instead, the resources that are scheduled to a process or task will be increased. Influencing the scheduler of the resources could be realized by the following two methods:

1. Affecting the scheduling of the resource that causes the degradation of the QoS support. As an example, consider two applications, our telephony application with its specific requirements to the characteristics of the communication link, and a best effort application that transfers data from or to the mobile device in order to synchronize the contents of a database or a calendar. If both applications run simultaneously on a mobile device that is currently connected via a 16 kbit/s link, the resources will be shared in a fair fashion, resulting in 8 kbit/s for each application. Assumed that the telephony application is a proactive one, it can ask the packet scheduler to get a higher data-rate for the communication link. This process is done as often as it is necessary for the telephony application to get its 13.2 kbit/s that ensures acceptable audio quality by using the GSM standard coding scheme. Given limited resources, increasing the resources for one application always comes together with further limiting resources for other applications. In the example, the data rate of the other application will be decreased to 2.8 kbit/s, which is reflected in a longer time period for carry out the synchronization of the database.
2. In some cases, an improvement of a resource will not be possible, as the current utilization of a resource reaches its capacity. In this case, a compensation of the limiting resource by exploiting other resources is conceivable. If, e.g., there is not enough bandwidth for a (proactive) application, it can ask for more computation time, that allows the usage of a more bandwidth-efficient, but also more complex compression algorithm to reduce the bandwidth requirements.

Thus, proactive applications are predestined for mobile environments with rather limited resources and unpredicted variations in the quality of the communication characteristics.

3. User Interface and Feedback Architecture

This chapter presents the main parts of our implementation, the simple user interface (Q-Button) and the layered feedback architecture for resource control.

3.1. Q-Button

Beside adaptive and proactive applications, the consideration of the user's preferences is an indispensable necessity for supporting QoS within the end-system, as only the user is able to decide which application is important for him/her and should be preferred. One solution that covers this aspect could be a mapping of the users' preferences to a set of application-specific QoS parameters as mathematically described in [11]. This mapping approach arises several problems. On the one hand, it is quite difficult to map user's preferences to QoS parameters as they depend mainly on the type of the application (e.g., frame-rate for a video application, sample-rate for an audio player). On the other hand, the preferences vary from user to user.

Thus, a uniformed, intuitive, and easy-to-use interface to the user is needed for controlling the current QoS support of an application. Especially in mobile environments, the easy-to-use property plays an important role, as a minimum of necessary user interaction is essential for the use in, e.g., cars, where the user has to concentrate on the traffic. Another aspect that effects the design of the user interface is the implementation of this interface given by the mobile devices itself, e.g. in the case of PDAs and handheld devices. Those devices usually have a restricted display size, so the user interface has to be simple and minimalistic.

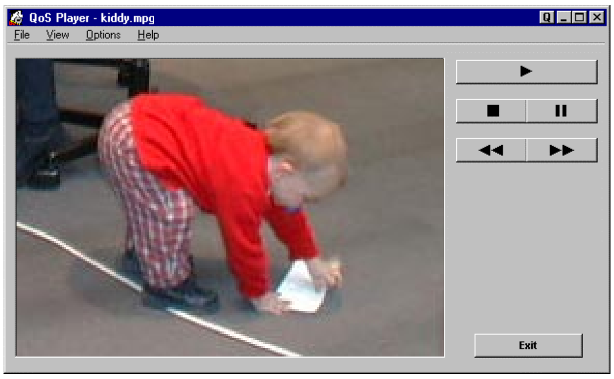

Figure 2. The Q-Button

In order to bring these requirements together, we use one single parameter that reflects the satisfaction of the user for an application. The user expresses his or her preferences by just clicking a simple button related to the application, in order to receive a "better" QoS support for this application. Figure 2 shows a draft of this simple interface: A button labeled with the letter Q (denoting Quality of Service) is integrated into the window title bar of each application started on the system. This Q-Button can be integrated into any windowing system (X-Windows as well as Windows CE) and does not necessarily depend on an application. It is the job of the operating system (and, if supported, of the application itself) to determine the meaning of "better". The user is not concerned with technical details and parameters detaching him or her from productive work.

The simple, but in many cases sufficient reduction to a "better/worse"-parameter brings up many advantages for common, adaptive, and proactive applications. The integration of the Q-Button in the windowing system allows the user to affect the quality of legacy and adaptive applications without their knowledge by influencing the scheduling of resources. Proactive applications can additionally react to the clicking by changing their internal (and thus monitored) QoS parameters. An example makes this interaction more clear. If the user initializes a video application with a desired frame-rate of 15 fps, an adaptive application tries to keep this frame rate, even if the user wishes an improved quality by clicking the Q-Button. However, proactive applications can react to the clicking of the Q-Button by modifying the internal parameters, e.g., by increasing the frame rate with each click. In order to meet the new QoS requirements (frame-rate), the application itself influences the scheduling of the resources. The realization of this interaction is described in more detail in section 3.3. The Q-Button is integrated in our layered feedback architecture, which is described in the next section.

3.2. Feedback Architecture for Realizing QoS Support

The feedback architecture concerned with the sharing of resources according to the user's preferences is shown in Figure 3. This architecture offers a unified interface for the applications to signal their current preferences. Clicking the Q-Button results in a signal passed to the interface. Note that proactive applications trying to influence the resource scheduling use this interface as well. We call these signals nag-signals because the user or the proactive application, respectively, is nagging at the given QoS. Our architecture further consists of a set of hierarchical resource schedulers and associated Monitoring&Control-entities. The monitoring part of these entities provides information about the current scheduling and utilization of the respective resource, whereas the control part influences the scheduling parameters. The resources currently being considered are the CPU time dispatched by the process scheduler, the network bandwidth dispatched by the data scheduler, and finally the current medium assigned by the network switching unit.

In the case of the process scheduler the scheduling parameters can be influenced in favor of a process. Clicking on the Q-Button then influences the share of computation time in favor of the related process. In the case of a simple application this might be enough to improve its performance, for example given an MPEG-player which needs more calculation time for encoding the frames. The feedback is therefore provided via the visible improvement of the applications performance. Modifying the scheduling parameters is done by the control part of the Monitoring&Control-entity. The monitoring part of this entity observes the time slices used by the process. If the process does not use up its time slice, this indicates that the nag-signal received via the system call denotes a problem that cannot be solved on this resource level. The nag-signal is then passed to the Monitoring&Control-entity of the networking subsystem. Note that there are different timing requirements on each level of the hierarchy. Applications on top of this architecture vary their requirements in the range of seconds without user interaction. While process scheduling operates in the range of milliseconds, the scheduling of packets in the networking subsystem even needs a higher granularity in time, usually in the range of nanoseconds.

It is obvious that our architecture does not increase the performance of the whole system. If there are not enough resources for an application (such as the computation time), the quality of the application will decrease. Instead, our architecture allows a QoS-based sharing of the available resources, which is not possible in common operating systems, such as Linux or Windows.

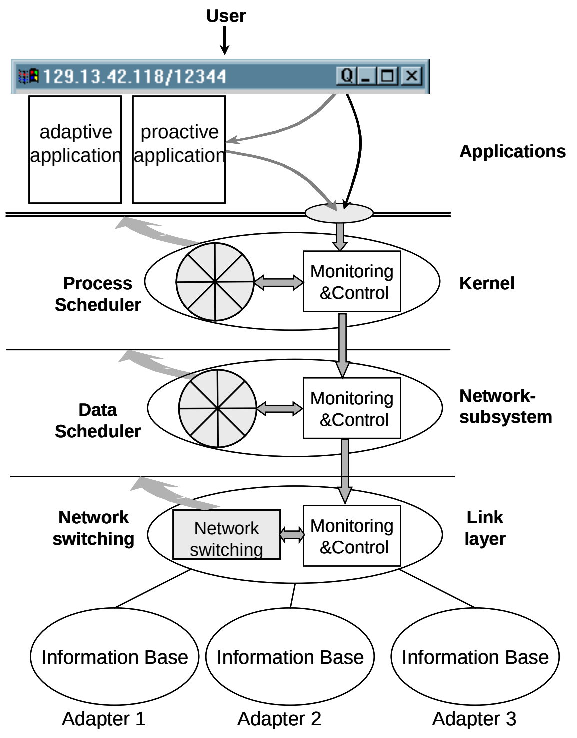

Figure 3. Feedback-based architecture of hierarchical resource schedulers

The data scheduler performs the shaping of the outgoing traffic. Therefore it holds context information about the traffic parameters of the respective flows. Based on the traffic parameters of the flows, the software-based data scheduler determines which packet (or cell in the case of wireless ATM) has to be sent at what point of time. Details for the software-based data scheduler are given in [12] and [13]. The control part of the Monitoring&Control-entity influences the data scheduler in favor of the data flow of the application related to the nag signal. Depending on the traffic parameters, different actions are performed by the data scheduler: For example, given a best effort or UBR traffic, the link share will be influenced in favor of this traffic, thus resulting in a better throughput. In the case of premium service or CBR traffic, the data scheduler keeps this connection and, e.g., in the case of a sudden network congestion or overbooking, slows down or even terminates other flows. First results using premium service within the end-systems are shown in [14]. The monitoring part of the Monitoring&Control-entity gains information about the performance of the traffic shaping by monitoring the output buffer. If there is a permanent buffer overflow, then the network capacity obviously is not sufficient. The monitoring entity then passes again a nag signal down to the Monitoring&Control-entity next in the hierarchy. In the case of the data scheduler the feedback loop is realized via the data buffers of the sending applications. If the output buffer is filled, the data scheduler can not pass the data of the sending applications to the network. Therefore the socket buffers of the sending applications fill and the applications thus notice that they cannot send data at the moment.

The Monitoring&Control-entity lowest in hierarchy is associated to the network switching at link layer. At this level the switching of the physical transmission path is done. Therefore the monitoring part of the entity has access to the information bases of the adapters connected to the different networks (e.g., GSM, WaveLAN, DECT etc.). These information bases contain quantitative and qualitative parameters. Typically these parameters are gained from physical parameters and measurements of the wireless link between mobile device and base station. Examples are the link status (up/down) or the strength of the carrier signal. The control part reacts to a nag signal passed down from the higher feedback layer. According to some implemented policies the control part issues the switching of the transmission path. Obviously a hysteresis will be part of such a policy, in order to avoid fast hopping between transmission links. In commercially used networks pricing aspects will also have heavy impact on the decisions of the control unit. These problems are addressed in the context of our current research in the UNIQuE project [15]. The handling of seamless data transport using multiple network adapters is addressed in, e.g., the MosquitoNet project [16] and not further discussed here.

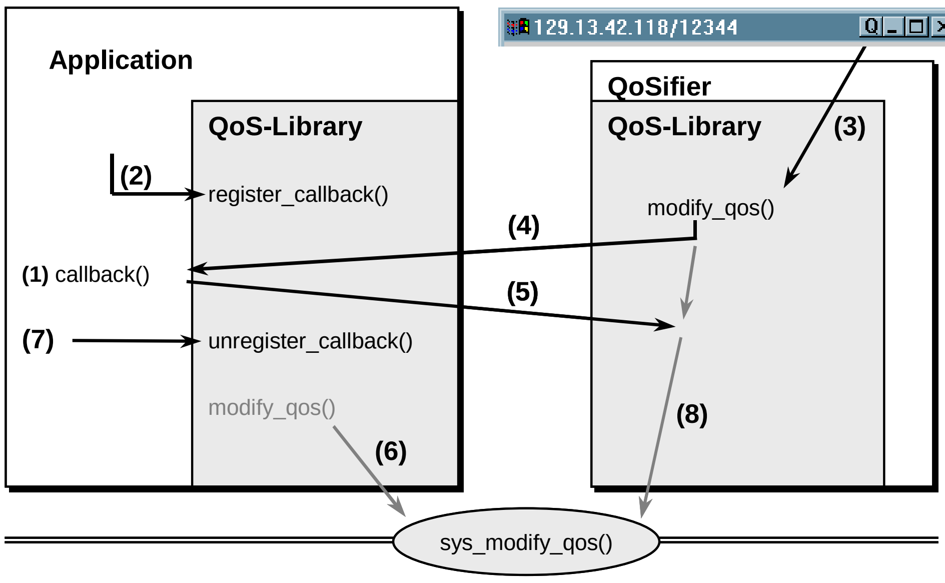

Figure 4. Feedback-based architecture of hierarchical resource schedulers

3.3. Implementation of proactive applications

For the implementation of our architecture described above, we use the Linux operating system as the kernel sources are freely available. In order to achieve a dynamic sharing of the computation time, our scheduler uses a flexible time-slice scheme. (The longer the time-slice assigned to a process, the higher the share of the available computation time.) This scheduler is combined with a modified mechanism for preempting processes that cares for a smoother utilization of the scheduled CPU time. The sharing can be influenced by the system call sys_modify_qos() shown in Figure 4. This unified interface allows for the realization of proactive applications in an easy way. The steps a proactive application has to perform are also illustrated in Figure 4.

The application has to implement a callback function (1). This function is responsible for updating the application-specific QoS parameters (e.g., frame-rate of a video player) as well as adapting the arguments of the modify_qos() call (that triggers this callback function) to achieve a more efficient adaptation process. This function will be called as soon as the application is notified. In the second step, this callback function must be registered by the application via the register_callback() procedure (2). The callback() function (1) has to be handed over as parameter. Before the application exits, unregister_callback() must be called for freeing the allocated resources (7).

If the user clicks the Q-Button, the modify_qos() function of our QoS-Library will be called (3) with three parameters: the process identifier of the (proactive) application that should be affected, a flag indicating whether the user requests QoS support for the application or not, and the default parameter that specifies the degree of improvement of the QoS support. In step (4), the QoS-Library tries to notify the specified application with the help of the process identifier. If the proactive application has performed the necessary steps described above, this notification leads to a confirmation (5) and a call of the callback function (1) to adapt the QoS parameters of the application (depending on the default parameter). Thus, the proactive application uses the system call (6) in order to be favored by the scheduler to achieve the adapted QoS parameters. The arguments of this system call are the same as used for the modify_qos() function (3). If the application is not a proactive one, the notification fails (steps (4) and (5)) and the system call will be triggered with the default value (8). Thus, our mechanisms can also be used for common and adaptive applications to achieve a better quality. In order to realize a stable QoS support, a proactive application can use the feedback mechanism without user intervention to meet its QoS requirements. If, e.g., a proactive video player cannot meet the specified frame-rate, it is able to run several times through the feedback loop (by directly calling modify_qos() from the QoS-Library) to achieve the specified frame-rate.

4. Performance Results

In order to evaluate new concepts, to run test scenarios, and to integrate advanced communication hardware, the UNIQuE research group installed a flexible, heterogeneous, and open testbed [15]. On the one hand, this testbed allows us to examine the integration of common end-systems into a DiffServ network. On the other hand, the UNIQuE testbed comprises a gateway to the wireless domain. This gateway is not only used for simple interconnection of the wireless network with the wired network. Furthermore, it realizes a Foreign Agent as useful for mobile IP [17] and acts as proxy or cache as needed for different TCP extensions developed for the wireless domain (I-TCP [18], snoop proxy [19], M-TCP [20]). Motivated by the traffic application scenario described in the beginning, we will focus here on the following research areas:

- *Handover:* Today’s mobile phones already perform handover between different base stations within the same system. New generations will also allow to change the network technology without interruption. One research topic is the provision of QoS for different networks and during handover. Communication systems and applications have to adapt to different bandwidths and delays. Furthermore, the networks have to reroute packets, change resource reservations for QoS provisioning, and perform authentication for security relevant communication scenarios. The UNIQuE architecture comprises a proxy and end systems that can switch between different wireless network technologies while still maintaining, e.g., TCP connections with a certain QoS.
- *WAP – Wireless Application Protocol:* The Wireless Application Protocol (WAP [5]) comprises protocols (Transport, Session, Security), application environments, telephony integration, and a markup language (Wireless Markup Language, WML) for wireless and mobile devices. The UNIQuE environment is an ideal base for experiments using this new architecture in addition to the evolutionary approach with TCP/IP and traditional browsers, HTML, and HTTP. Research topics are the interworking of WAP and Internet protocols, gateways for content transformation (HTML to WML), and transaction oriented applications on top of WAP. As WAP tries to cover the differences between different transport technologies (GSM, CDPD, UMTS etc.) it is the ideal framework for a scenario with multiple network interfaces and frequent changes of access points to the fixed network.
- *QoS on mobile end-systems:* While much work providing QoS to applications focussed on high-performance “wired” workstations, our project tries to provide QoS on mobile, possibly low-performance end-systems. This includes protocol support, operating systems, and applications. Protocols for mobile systems include mobile IP and wireless ATM, as well as GSM, DAB, and new protocols for the UMTS. Even small mobile phones require operating systems, but these have to be implemented under certain restrictions (power, memory requirements, processor speed). However, users still want certain service guarantees for applications including audio, (low-quality) video, or web browsing. Not only operating systems, but also applications have to be adapted to the mobile environment.

Two measurements show the features of our approach. First, we have a look at the effects of modifying application-specific QoS parameters of a typical multimedia application. This test scenario demonstrates that applications can support QoS parameters, which can be dynamically changed by just clicking on the Q-Button. In our second scenario, we compare a proactive application and a standard application to show the benefits of using our approach to meet QoS requirements of an application. This example shows that proactive applications are able to compensate for a long-term degradation of the QoS due to a limited resource. In order to visualize the possibilities of our architecture, we modified an existing MPEG-player in a way that it allows the interaction with the Q-Button. If the user clicks on the Q-Button, the frame-rate of the video is increased by 2 fps; if the user presses the shift key while clicking the frame-rate will be reduced by 2 fps.

The result of the first test is shown in Figure 5. The player was started to display a video stream with a rate of 22 fps. After t = 10 s, t = 20 s, and t = 30 s, the Q-Button was pressed to increase the frame-rate for 2 fps each time. When playing the video with a rate of 28 fps in the time interval [30 s; 40 s], the load on our test-system reaches the limit of the current resources, which manifests in heavy variations of the frame-rate. At this point, proactive applications would try to meet its QoS requirements by requesting more computation time from the OS. At t = 40 s the Q-Button was clicked twice (while pressing the shift key) to reduce the frame-rate to 24 fps.

The second test aims at our mechanisms for supporting QoS by the interaction with the operating system. This scenario shows that a proactive application is able to support stable QoS parameters in environments with varying availability of resources. We implemented a proactive MPEG-player (based on the MPEG-player described above) which works as follows. The player is started with a pre-determined frame-rate which can be modified by clicking the Q-Button and which is monitored within the application. If the current frame-rate falls below a threshold, the player notifies the operating system to get more resources – without any user interaction. This process will be iterated until the specified threshold is reached. In the example given in Figure 6 the proactive application has to compete for the available computation time with the standard MPEG-player described in the first scenario (the standard player does not make use of the QoS mechanism offered by our architecture).

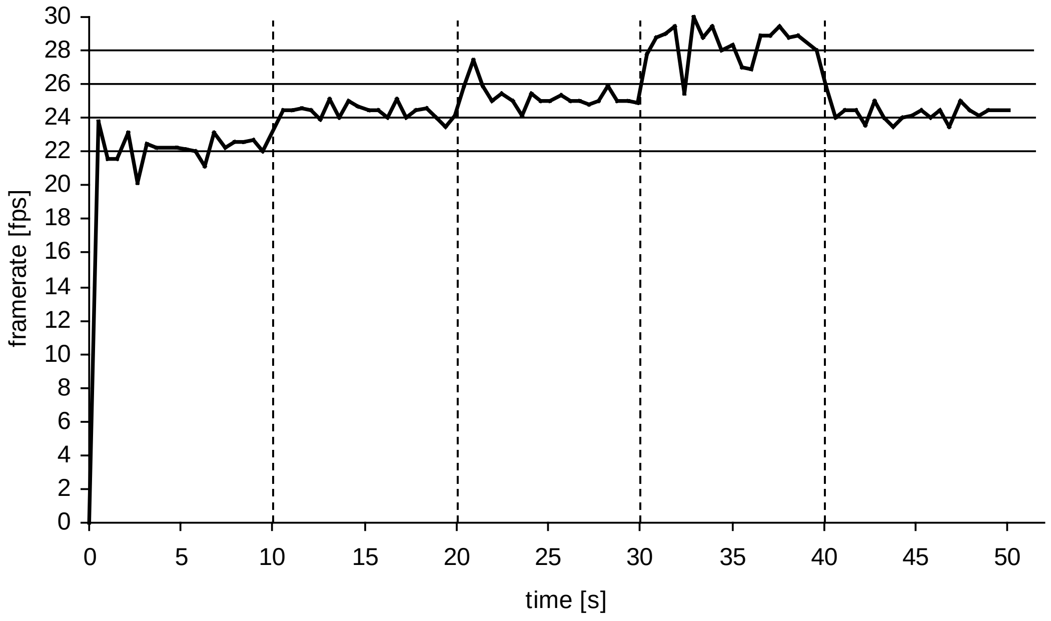

Figure 5. Modification of application-specific QoS parameters via the Q-Button

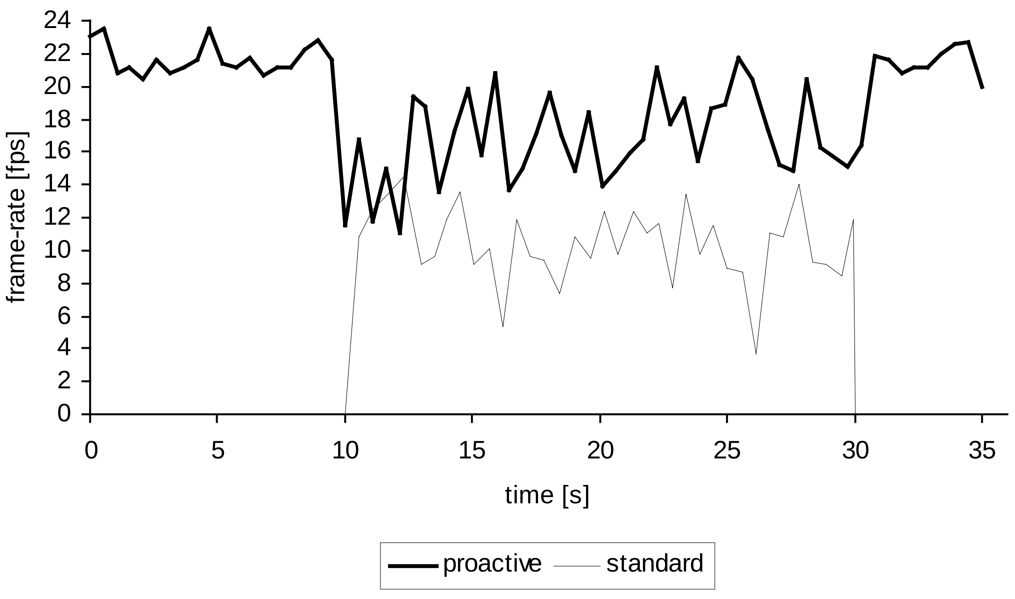

Figure 6. Proactive versus standard application

The proactive MPEG player was started at t = 0 s with a pre-determined frame-rate of 22 fps. If the frame-rate falls below the threshold of 20 fps, the player asks the operating system for a better QoS support. After 10 seconds, the standard player was started to display a video stream with the best possible frame-rate. As can be seen, the frame-rate of the proactive application drops to the same level as that of the non-proactive one (about 13 fps). Now, both applications compete for the resources and each application gets about half of them. But then, the frame-rate of the proactive MPEG-player automatically increases, while the frame-rate of the original MPEG player decreases slightly.

In order to use our approach in mobile and wireless environments, our architecture has to take care of the scarce resources available on mobile devices. Therefore, we avoid active waiting and additional processes or threads to save memory and CPU time for the communication between applications and the QoS-Library. Instead, we use asynchronous notification and named pipes in step (4) and passive waiting on kernel messages for the response from the application in step (5) in Figure 4. Our QoS-Library has a tiny size of 2.7 kbyte and, thus, complies with the restricted memory size in mobile devices.

5. Conclusions

Driven by the demands of our application scenario – road traffic with heterogeneous wireless networking technologies – we believe in the need for adaptive and proactive applications to support mobility. We presented a simple and intuitive user interface to affect the QoS of an application which can be used without any knowledge of internal, application-specific QoS parameters. The flexible architecture allows for an automatic adaptation to different network characteristics, furthermore, proactive applications can influence the behavior of the CPU scheduler via a unified interface in a simple way.

Future work will comprise the integration of our approach in larger-scale experiments within our UNIQuE testbed using different types of mobile end-systems and experiments with end-to-end quality of service using the differentiated services approach together with mobile IP (maintaining QoS and adapting while roaming from one network to another).

6. References

[1] Infrared Data Association, http://www.irda.org/, 1999

[2] Bluetooth consortium, http://www.bluetooth.com/, 1999

[3] K. Nichols, S. Blake, F. Baker, D. Black: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, December 1998

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An Architecture for Differentiated Services, RFC 2475, December 1998

[5] Wireless Application Protocol Forum, http://www.wapforum.org/, 1999

[6] ETSI, Standards for DAB, DECT, GSM, and UTMS, European Telecommunications Standards Institute, http://www.etsi.org/, 1999

[7] Lucent, WaveLAN II, Lucent Corporation, http://www.lucent.com/, 1999

[8] Lakshman, AQUA: An Adaptive Quality of Service Architecture for Distributed Multimedia Applications, Dissertation, University of Kentucky, 1997

[9] T. Roscoe, The Structure of a Multi-Service Operating System, Dissertation, Queen's College, University of Cambridge, April 1995

[10] B. Li, Adaptive Behaviour of Quality of Service in Distributed Multimedia Systems, Dissertation, University of Illinois, 1997

[11] A. Richards, G. Rogers, M. Antoniades, V. Witana, Mapping User level QoS from a Single Parameter, *Proceedings of the 2nd International Conference on Multimedia Networks and Services (MMNS'98),* November 1998

[12] J. Schiller, P. Gunningberg, Feasibility of a Software-Based ATM Cell-Level Scheduler with Advanced Shaping, *Broadband Communications '98 (BC'98),* Stuttgart, Germany, 1998

[13] J. Schiller, Feedback controlled scheduling for QoS in communication systems, *IFIP Conference on High Performance Networking (HPN'98),* Vienna, Austria, 1998

[14] M. Bechler, H. Ritter, J. Schiller, Integration of a Traffic Conditioner for Differentiated Services in End-Systems via Feedback-Loops, *Broadband Communications '99 (BC'99),* Hong Kong, China, 1999

[15] UNIQuE, Universal Network Infrastructure with Quality of service Enhancements, http://www.telematik.informatik.uni-karlsruhe.de/forschung/UNIQuE/, 1999

[16] X. Zhao, C. Castelluccia, M. Baker, Flexible Network Support for Mobility, *Proceedings ACM/IEEE MobiCom'98,* 1998

[17] C. Perkins, IP Mobility Support, RFC 2002, October 1996

[18] A. Bakre, B. Badrinath: I-TCP, Indirect TCP for Mobile Hosts, *Proceedings 15th International Conference on Distributed Computing Systems (ICDCS),* Vancouver, 1995

[19] E.A. Brewer, R.H. Katz, Y. Chawathe, S.D. Gribble, T. Hodes, G. Nguyen, M. Stemm, T. Henderson, E. Amit, H. Balakrishnan, A. Fox, V. Padmanabhan, S. Seshan, A Network Architecture for Heterogeneous Mobile Computing, *IEEE Personal Communications*, vol. 5, no. 5, 1998

[20] K. Brown, S. Singh: M-TCP, TCP for Mobile Cellular Networks, *ACM Computer Communications Review*, vol. 27, no. 5, 1997