A Resource Management Architecture for Mobile Satellite-based Communication Systems

Philipp Drieß, Florian Evers, Markus Brückner
Integrated Communication Systems Group
Ilmenau University of Technology
Helmholtzplatz 5
98693 Ilmenau, Germany
Email: {philipp.driess, florian.evers, markus.brueckner}@tu-ilmenau.de

***Abstract*—Today's Quality-of-Service (QoS) architectures for packet switched networks, especially reservation-based architectures such as IntServ, strongly depend on transmission systems with link characteristics that do not change over time. In wireless network access technologies however, this requirement makes implementation of QoS challenging as links change due to effects such as shadowing or fading caused by node mobility. In this paper, a novel cross-layer reservation-based QoS system is presented. It is able to deal with changing link conditions and notifies the affected applications with the help of feedback messages. This allows graceful degradation in compliance with the requirements of the applications. The architecture presented in this paper focuses on a satellite-based network for rescue teams during a disaster scenario. Based on a geostationary satellite and highly mobile ground stations, this network is characterized by long transmission delays and small, changing link capacities. The system offers a resource reservation scheme on the physical layer which is integrated with the approach presented here to design a cross-layer resource management architecture. This results in a reservation system for high-latency, low-capacity and unstable links.**

***Index Terms*—quality of service; satellite communication; mobile communication; IntServ; signaling**

I. Introduction

In contrast to wire-based transmission systems such as Ethernet, the transmission conditions of wireless systems are considered *unstable*. This is especially true if nodes become mobile: laptops that are carried around experience different kinds of fading effects, and mobile satellite terminals are constantly affected by trees, clouds and other obstacles that impair the line-of-sight transmission to the satellite.

Such links with changing conditions can not be avoided, which leads to problems regarding support for Quality-of-Service (QoS). Reservation-based schemes like "Integrated Services" (IntServ [1]) depend on networks with *stable* links for their capacity management, which fails if the available capacity is a dynamic parameter. In contrast, "Differentiated Services" (DiffServ [2]), an architecture that is based on the differentation of traffic into classes with specific properties, does not offer reservations at all and thus is not able to offer guarantees to the applications.

Depending on the intended use case having guarantees might be a requirement. In the research project "Mobile Satellite Communications in Ka-Band" (MoSaKa, see [3] for an introduction), a satellite-based communication system is developed to support rescue teams in disaster scenarios. In such a system, voice communication is one very important application, ideally in combination with video. As satellite resources are scarce, not all communication attempts can be admitted. However, continuous communication streams like voice conversations are not the only traffic in disaster communication systems. They are also used for all kinds of data traffic like digital maps, status reports, and position information. Therefore, a packet-switched approach based on a protocol suite like TCP/IP is the most flexible approach to build such a network. To serve important applications like voice reliably, a reservation-based scheme that assures QoS can be implemented, causing the aforementioned problem regarding the unstable characteristics of the link to the satellite. The availability of a QoS system coping with those limitations will be a key factor for being able to use packet-based satellite communication systems as backbones, especially in disaster scenarios.

In this paper the MoSaKa QoS system, a novel reservation-based QoS architecture that is able to cope with unstable links, is presented. The focus is on satellite-based networks. This empowers rescue teams to communicate in environments without communication infrastructure, which is the research area the MoSaKa project is looking at. However, the algorithms shown in this paper could also be applied to other QoS-enabled wireless transmission systems, such as IEEE 802.11e.

The remainder of this paper is organized as follows: Section II describes the environment for which the presented solution was designed. From this the system requirements are derived in Section III. Section IV gives an overview over the related work. Following this Sections V and VI present the proposed architecture and the test environment which is being built to evaluate the approach. The paper finishes with an outlook to future work and a conclusion.

II. The MoSaKa Research Project

The MoSaKa project funding the research presented in this paper aims at developing a complete satellite communication stack from the antennas up to the QoS management, including antenna tracking systems and a decentralized resource allocation scheme. In this paper, the main focus is on the QoS system as it is seen by the higher layers. Layer 2 and below are

 ISBN: 978-1-61208-199-1

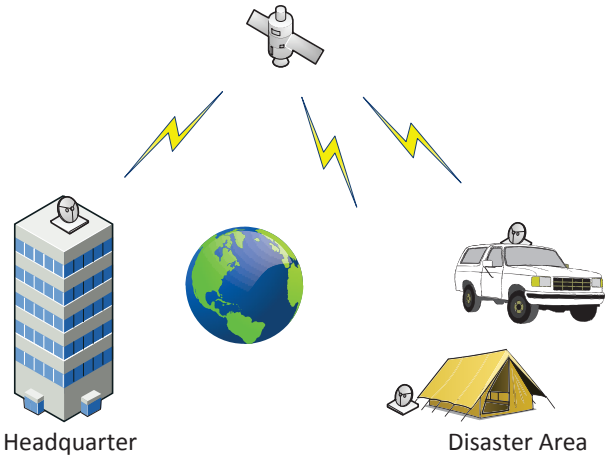

Fig. 1. Typical use cases for MoSaKa entities: fixed, nomadic and mobile terminals

only mentioned insofar as they are required to explain design decisions taken in the higher layers.

Figure 1 shows a typical usage scenario: some nomadic and mobile terminals deployed in a remote area use the satellite link to communicate with their headquarter. Each terminal uses a dynamic set of services resulting in individual traffic demands. The communication link from each terminal to the satellite is considered unstable: the link quality fluctuates with the movement of the terminal and changing environment conditions. Most of these fluctuations are short, but some may persist for a longer time span.

Large-scale disaster recovery operations cause a huge demand for communication. Satellite links are a – comparatively – scarce resource with only a small capacity and long delays (for geostationary orbits at a height of $\approx 36\,000$ km the time-of-flight is already more than 100 ms for one direction). These two effects have to be considered if a system-wide QoS infrastructure is implemented.

III. Requirements

Based on the scenario given in Section II, a set of requirements that a QoS infrastructure has to fulfill can be derived.

Efficient handshakes

The main issue in designing an efficient signalization scheme for QoS requirements is the long transmission delay introduced by the satellite link. With a round trip time of ≈ 400 ms, complex handshakes with multiple messages travelling back and forth are imposing an unacceptable overall latency.

Efficient handling of link instability

Today's QoS systems assume stable links with static resources that can be utilized for reservations. This assumption is no longer valid in mobile, satellite-based communication systems or even in mobile communication systems in general. Over the time, the propagation conditions are subject to change. The QoS infrastructure presented in this paper must be able to cope with unstable link conditions.

Cross-layer link usage optimization

Satellite-based communication with multiple terminals takes place on a shared broadcast medium. To enable parallel transmissions via one single satellite the MoSaKa physical and MAC layers have to distribute the available link spectrum to all terminals that compete for resources. This happens with respect to the individual resource demands of each terminal. Beforehand these resource demands have to be derived from higher-layer QoS requirements that originate from the applications.

Due to the long delay of the broadcast medium the resource assignment procedure takes place in a distributed manner without central coordination and without any point-to-point negotiation. If the link share decreases the higher layer reservations may not fit anymore. In that case, the QoS system has to evaluate all admitted reservations based on their properties to keep as many of them as possible active.

A resource management system suitable for mobile satellite communication has to address those requirements. Existing solutions fall short in one or the other aspect prompting the development of a new architecture for the MoSaKa project.

IV. Related work

QoS architectures such as IntServ [1][4][5] or DiffServ [2][6] are well known and have a wide range of acceptance. Nevertheless, they have a variety of issues regarding unstable link conditions.

IntServ

IntServ is an architecture that offers hard guarantees regarding QoS parameters. Applications request reservations via a signaling protocol such as the "Resource Reservation Protocol" (RSVP [7]) or "Next Steps in Signaling" (NSIS [8][9][10]) to announce their individual traffic requirements. On each node along the transmission path an IntServ entity manages and monitors the traffic regarding the requested resources.

Applying IntServ upon an unstable link leads to problems if the link capacity starts decreasing. This results in a situation where the sum of all accepted reservations does not fit into the link budget anymore and the reservations are violated. As no feedback mechanism is available, the system has to withdraw reservations. Affected applications can only deal with this situation by reserving a new path with different parameters or ceasing communication altogether. Signaling new paths causes additional message load on the already limited link, contributing further to the congestion.

DiffServ

One of the problems of IntServ in large-scale networks is its bad scalability. Due to the state kept in every intermediate node, IntServ installations do not scale to Internet-sized networks. This prompted the development of DiffServ which is based on differentiation of traffic into classes which can be treated differently by the network. This allows the assignment of transmission priorities to distinguish different types of traffic.

 ISBN: 978-1-61208-199-1

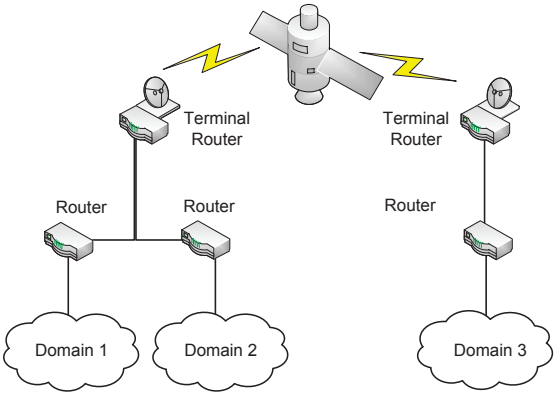

Fig. 2. The network of a scenario where the MoSaKa QoS architecture is deployed

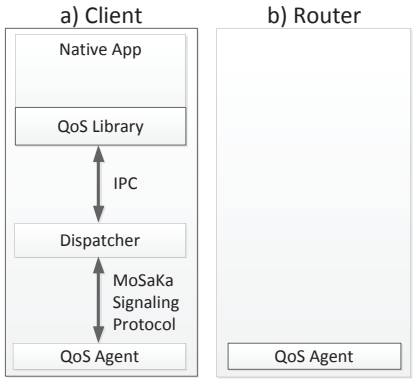

Fig. 3. Two kinds of nodes exists in the MoSaKa network: clients and routers.

As DiffServ does not support the reservation of a communication path, it does not offer any guarantees. Excessive traffic in a single class might exceed the link capacity, causing packet loss for all affected applications.

For use in mobile satellite environments, both approaches lack certain desirable features. This prompted the development of a new approach based on IntServ. While the scalability of IntServ to large networks might be a problem, this will not be an issue in the system at hand. However, the guaranteed reservation of communication paths as offered by a reservation-based system is crucial in a disaster scenario. MoSaKa aims at solving the challenges arising from unstable links while keeping the reservation features of IntServ.

V. The QoS architecture of MoSaKa

One part of the MoSaKa project has the goal to build a reservation-based QoS architecture that is able to cope with unstable link conditions. Therefore, an IntServ-like approach was chosen, which introduces management entities on each intermediate node as well as on each end system. These entities are aware of all reservations that pass the respective node. In the depicted scenario, static routing in the backhaul is assumed, which ensures that each packet of a flow always takes the same route through the network.

The resulting topology is depicted in Figure 2. The central component is the satellite-based communication system with a geostationary satellite and multiple terminals as ground stations. The satellite link is considered to be a bottleneck with high latency. The terminals act as routers for the IP protocol, and attach local networks to the satellite network.

The result is a QoS architecture without a central coordinator. Unfortunately, this approach inherits two issues of IntServ: it has scalability problems and might fail if the links are unstable. The former can be neglected with the depicted use case in mind, but the latter will be discussed in this paper.

A. Software components

The components introduced by the MoSaKa QoS architecture are depicted in Figure 3. There are two main components: the QoS Agent and the Dispatcher.

1) The QoS Agent: The QoS Agent is a management entity that exists on each node of the network including routers and clients. This entity is aware of all ongoing reservations that pass the node and has an overview of the transmission resources of each interface that the node possesses. This allows the QoS Agent to decide whether a subsequent reservation can be admitted or has to be rejected. For the purpose of transmitting reservation request, a signaling protocol such as RSVP or NSIS is required. The QoS Agent intercepts protocol messages and interprets them as necessary. On the satellite terminal, it also communicates with the lower layers to detect if the link deteriorates.

QoS components like traffic metering and shaping are highly dependent on the underlying operating system of a node. It is the task of the QoS Agent to adapt the high-level reservations to the QoS primitives available on the host to allow a wide deployment of the architecture in heterogeneous networks. Each agent, therefore, consists of a generic part handling the signaling and admission control and a system-specific part configuring the underlying operating system services.

2) The Dispatcher: The Dispatcher is an optional component that is only required if a given node has applications running on it, making it a client. A dispatcher acts as a broker between the applications running on the client and the QoS system in the network. The applications talk to the Dispatcher using “interprocess communication” (IPC). The Dispatcher handles all QoS-related interaction with the network relieving the application from doing so. Additionally it serves as an entry point for requests and notifications from the network, decoupling the local application structure and the state saved along the communication path. From the network point of view the Dispatcher is the entity that holds a reservation and renews it as necessary.

Reservations are always triggered by an application. The Dispatcher merely acts as a proxy. Therefore applications are a part of the MoSaKa QoS architecture as well and need to be modified to take full advantage of the system. One has to distinguish QoS-enabled applications, legacy applications and

 ISBN: 978-1-61208-199-1

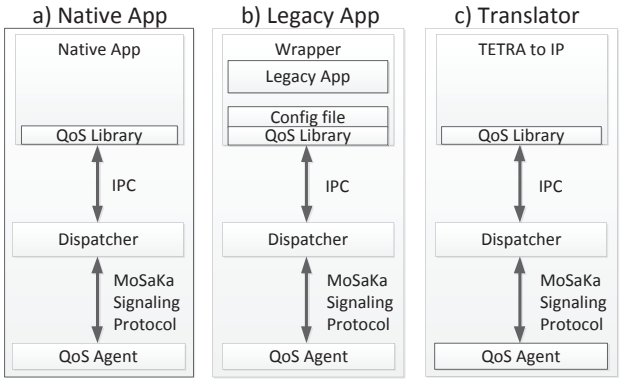

Fig. 4. Three kinds of applications are distinguished: Native, legacy and translator applications.

translator applications.

QoS-enabled applications: As shown in Figure 4 a, a QoS-enabled architecture includes the MoSaKa QoS library. This library offers an high-level API to interact with the QoS architecture and allows the programmer to easily request transmission resources or to be notified if an active reservation fails or deteriorates. Such an application is aware of its traffic demands and is able to request the appropriate amount of resources before it starts transmitting. Additionally, it is aware of the fact that feedback messages may arrive that indicate that the reservation is affected by the current transmission conditions.

Legacy applications: All IP-based applications that exist today are considered as legacy applications. They are not aware of an API to request transmission resources, resulting in traffic that is not known to the local QoS Agent. Two approaches are possible: this traffic can be considered as "best effort" traffic, which may or may not pass a bottleneck in the network, or it can be reserved with the help of a "wrapper application" (Figure 4 b).

Such a wrapper loads a predefined set of QoS requirements from a configuration file, initiates the reservation process and then, if successful, executes the legacy application. In that case, the application does not need to know anything about the QoS system, but benefits from it nevertheless, as the required resources are reserved.

The reservation is held all the time even if the application does not emit traffic. Even worse, to initiate a reservation, the endpoint must be named, which limits the application to a given set of predefined peers. However, such a wrapper can be seen as an intermediate solution until the affected applications implement the QoS scheme.

Translator applications: As a third kind of applications, a translator application, acts as a gateway to other kinds of reservation schemes or networks such as circuit-switched telephony systems. Such an entity is a special case of a QoS-enabled application (Figure 4 c).

In disaster scenarios, connections with other network types such as "Terrestrial Trunked Radio" (TETRA [11]) may be required. A dedicated gateway node with a TETRA base station and a translator application installed on it can interconnect both networks, allowing TETRA terminals to make telephone calls to the headquarter via the satellite. The translator application is aware of the required resources of a TETRA channel, as the traffic requirements of the codecs involved are known. This allows it to send suitable reservation requests into the MoSaKa network.

B. The QoS-enabled network

The network consists of two kinds of nodes: intermediate nodes are referred to as routers, and end systems are referred to as clients.

As routers have no applications running on them, the only entity required here is the QoS Agent. The routers that are connected to the satellite system are referred to as terminals. On such a terminal the QoS Agent is equipped with additional capabilities to manage the link to the satellite.

Clients, as they are considered as user equipment, have applications running on them additionally requiring the Dispatcher as a bridge to the network.

On each network node the QoS Agent has to configure the local packet forwarding entity of the operating system to stop misbehaving applications from congesting the outgoing interfaces. It should be impossible that traffic, that exceeds the capacity of the outgoing link, causes packet loss for flows that have been negotiated before. This is achieved relying on platform specific mechanisms to control traffic flow like "Traffic Control" (tc) and "Netfilter" on Linux or the MoSaKa MAC scheduler on the satellite terminal.

The signaling scheme

On each client QoS-enabled applications communicate with the local Dispatcher via an API offered by the MoSaKa QoS library. Through this API the application informs the QoS System about the amount of resources it requires for a transmission to a well-defined peer. The Dispatcher creates a reservation request signaling message that it sends to its peer entity, the Dispatcher on the destination node. All signaling messages are intercepted by each QoS Agent along the path to the destination, allowing them to decide whether to accept or to deny the reservation request. If such a reservation has to be denied because of insufficient remaining link resources, a negative acknowledgement is sent back to the initiating Dispatcher. This results in a deletion of the pending reservation on all intermediate nodes and leads to a negative acknowledgement to the application via the QoS Library.

In case of success, the QoS Agent on the destination node informs the local Dispatcher of an incoming reservation request. This Dispatcher may be aware of local applications as they are allowed to register to it beforehand. Nevertheless, it sends an acknowledgement back to the originator. This message is intercepted by all QoS Agents again and results in an orderly created reservation along the whole path.

 ISBN: 978-1-61208-199-1

C. Feedback mechanism

To deal with changing link conditions, MoSaKa adds a feedback mechanism to the signaling protocol. Such feedback messages originate from an QoS Agent observing a deteriorating link on its outgoing interfaces and are sent to all applications that hold reservations affected by this degradation.

On each node, the link hardware is monitored by the local QoS Agent. Additionally, this entity is aware of all active reservations that involve this link, allowing it to decide whether the remaining capacity is still high enough to serve all reservations. If the capacity falls below the amount of reservations, the QoS Agent starts to optimize. Optimization is done by building a set of allowed reservations starting from the one with the highest priority. Reservations are incrementally added to the set if there is still capacity available. This simple optimization algorithm is suited well for highly hierachical communication environments like desaster recovery operations.

After optimization the QoS Agent has a list of reservations that still have enough resources and a list of reservations that do not fit into the link anymore. Instead of cancelling these reservations as architectures such as IntServ would have to do, the QoS Agent of MoSaKa is able to put affected reservations "on hold". Such a reservation is still known to the whole path, but can not be utilized for the moment. If the link recovers shortly later, the reservation is reactivated by another feedback message. As the MoSaKa scenario states that link degradations are short in nature, this approach allows a reservation scheme with a low amount of signaling messages. Applications do not need to actively poll the network for free resources which would introduce a high signaling load. Nevertheless, if the link stays degraded for a longer time span, the QoS Agent may cancel the reservation to prevent congesting the network with reservations that cannot be served anyway.

In the MoSaKa QoS architecture, signaling messages are usually exchanged between Dispatchers on two peer nodes, and are intercepted by all QoS Agents on each intermediate node including the end nodes that run the Dispatchers. If a given reservation has to be suspended, the QoS Agent creates signaling messages and sends them to both Dispatchers, allowing all other QoS Agents to notice that this reservation is currently "on hold".

If a signaling message arrives at a Dispatcher, it relays it to the respective application which triggers a trap in the QoS library informing the application about an accepted, suspended, resumed or cancelled reservation.

This feedback scheme is new and allows a graceful degradation of communication. To underline this, one of the most important applications of the MoSaKa scenario is analyzed: Video chat.

D. Impact on Video chat

If a user starts a video chat session with the headquarter, the video chat application tries to reserve resources for the video data and for the audio data separately. As a video chat session is bidirectional, the reservation requests resources for both directions at the same time, allowing the signaling handshake to complete fully after just one round trip. If the reservation handshake is completed without rejections from intermediate systems the path is active and can be utilized.

If the satellite link deteriorates, this is noticed by the QoS Agents on the satellite terminals. They start the optimization process which results in the less important video streams to be put "on hold" to keep the audio streams active. A feedback message is sent to the Dispatchers at both ends of the path, resulting in the deactivation of the video stream in the application. If the link recovers, another feedback allows the video stream to be resumed. Applications may provide a visual indication based on the network state to make the proess transparent and increase user satisfaction. If the link fails to recover the reservation is cancelled by the network. This frees resources permanently for reuse by other applications.

The "on hold" state allows the system to bridge short link degradations that are common in mobile satellite communication without reconfiguring the whole path causing large signaling effort. The feedback mechanism allows applications to intelligently react to those changes in the network. Especially for satellite links with long delays and a low capacity, such a scheme is essential to operate as desired.

E. MoSaKa Satellite Terminals

To check whether all active reservations fit into the current link capacity, the QoS Agent has to obtain this information. For that purpose, technology-dependent functionality is required to interact with Ethernet, IEEE 802.11e or the MAC- and PHY layers of the MoSaKa satellite terminals.

The MoSaKa satellite link offers QoS-enabled lower layers. From the physical layer point of view the satellite link is always a shared medium. Each terminal can be received (although not necessarily decoded due to signal quality issues) by every other terminal via the satellite. Therefore it is necessary to allocate parts of the link spectrum to specific sender terminals to prevent collisions. To accommodate for changing link conditions this allocation is not static but takes place every $250\ \mathrm{ms}$. Each active terminal is assigned a short time slot on the lower layer (L2) signaling channel in this period and broadcasts its resource request to all other terminals. Based on this information, each terminal applies the same resource assignment procedure and comes up with the same resource allocation vector for the next $250\ \mathrm{ms}$ data transmission frame. A reservation on the lower layers is valid for only one slot, and has to be renewed continuously by using the L2 signaling channel.

If the link between one terminal and the satellite deteriorates, the QoS Agent on this terminal gets informed about a lower amount of transmission resources that this terminal got assigned for the next data transmission frame. This allows the QoS Agent to check whether high-level reservations and available link share still match and take appropriate actions if they do not.

VI. Evaluation

The presented architecture will be implemented as a proof-of-concept for the Linux operating system. To evaluate the

 ISBN: 978-1-61208-199-1

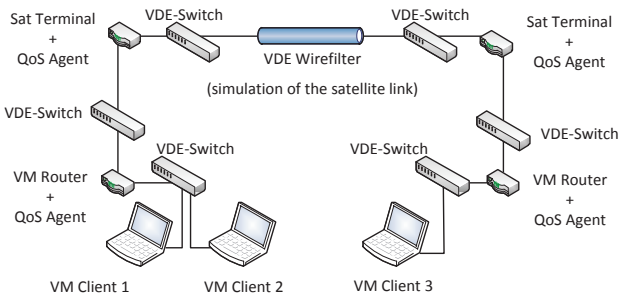

Fig. 5. The MoSaKa virtual testbed based on QEMU/KVM and VDE

implementation a virtual network environment based on QEMU/KVM [12] is currently set up. Different Linux-based guest systems are interconnected using "Virtual Distributed Ethernet" (VDE [13], by Virtual Square). Heart of the testbed is the satellite link emulated by wirefilter, a part of VDE that allows to set up virtual Ethernet connections with various parameters such as the data rate, delay or the packet loss rate. At each end of the emulated link resides a Linux-based guest system acting as a the MoSaKa satellite terminal. Connected to them are networks of different structure depending on the scenario. A possible scenario is shown in Figure 5. Each part of the network contains an additional QoS Agent as well as a set of clients acting as signaling and traffic sources.

In parallel to this virtual testbed designed to test the high-level parts of MoSaKa, the complete stack including the terminal hardware is currently being implemented. This "real world" setup will allow tests of the whole communication path developed during the project. The resulting architecture will include all layers, from the QoS-enabled applications down to the antenna. The system is based on a virtual satellite placed on a tower. A motion emulator and a channel simulator provide the infrastructure to simulate the influence of mobility on the Ka-band channel. A directional antenna on a 3D rotor acts as the mobile terminal. The motion emulator is driven by data from measurement campaigns carried out earlier to create a communication environment as it is expected "in the wild". Using this realistic emulation environment close-to-real measurement results on all layers are expected from this implementation.

VII. Conclusion and future work

This paper presented a novel QoS architecture called "MoSaKa QoS" for mobile satellite communication links. Existing approaches like IntServ are not suited for such an environment with unstable, long delay links. The optimistic, bi-directional signaling architecture of MoSaKa QoS with the included feedback mechanism supports reaction to link changes while minimizing the number of messages exchanged. Cross-layer resource optimization spanning layers 1 to 3 enables a better usage of the currently available channel, maximizing the user experience.

Future research should investigate into the possibilities opened up by the MoSaKa feedback mechanism. Modern audio and video codecs offer various output profiles with different data rates and quality settings. An integration with the QoS system might provide further options for graceful degradation of the link.

From the architecture point of view further research might look into alternative QoS models based on probability distributions instead of hard thresholds. Equally interesting are novel reservation models which provide more powerful ways to express requirements, enabling the system to adapt better to changing conditions without consulting the application. Another open question is the extension of the optimization algorithm to more general use cases without a strong hierarchy among communication paths.

At the moment MoSaKa uses a static IPv6 routing setup. In the future the system should adapt to various routing protocols to enable network-level node mobility.

One final research direction is the integration of legacy applications into the system. Applications that cannot be adapted to the MoSaKa system could be integrated using translator applications/application-level proxies or DiffServ-like classification approaches.

References

[1] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," *RFC 2210*, September 1997.

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

[3] M. Hein, A. Kraus, R. Stephan, C. Volmer, A. Heuberger, E. Eberlein, C. Keip, M. Mehnert, A. Mitschele-Thiel, P. Driess, and T. Volkert, "Perspectives for Mobile Satellite Communications in Ka-Band (MoSaKa)," in *EuCAP 2010: The 4th European Conference on Antennas and Propagation*, Barcelona, Spain, 04 2010.

[4] J. Wroclawski, "Specification of the Controlled-Load Network Element Service," *RFC 2211*, September 1997.

[5] S. Shenker, C. Partridge, and R. Guerin, "Specification of Guaranteed Quality of Services," *RFC 2212*, September 1997.

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

[7] A. Mankin, Ed., F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang, "Resource ReSerVation Protocol (RSVP) – Version 1 Applicability Statement Some Guidelines on Deployment," *RFC 2208*, September 1997.

[8] R. Hancock, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next Steps in Signaling (NSIS): Framework," *RFC 4080*, June 2005.

[9] J. Manner, G. Karagiannis, and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling," *RFC 5974*, October 2010.

[10] E. G. Ash, E. A. Bader, E. C. Kappler, and E. D. Oran, "QSPEC Template for the Quality-of-Service NSIS Signaling Layer Protocol (NSLP)," *RFC 5975*, October 2005.

[11] E. Re, M. Ruggieri, and G. Guidotti, "Integration of TETRA with Satellite Networks: A Contribution to the IMT-A Vision," *Wirel. Pers. Commun.*, vol. 45, pp. 559–568, June 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1363306.1363328

[12] F. Bellard, "QEMU, a fast and portable dynamic translator," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 41–41. [Online]. Available: http://dl.acm.org/citation.cfm?id=1247360.1247401

[13] R. Davoli, "VDE: Virtual Distributed Ethernet," in *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 213–220. [Online]. Available: http://dl.acm.org/citation.cfm?id=1042447.1043718

 ISBN: 978-1-61208-199-1