Technical Report 357/07

Mapping of formal Network Quality-of-Service Requirements

Christian Webel[1] Reinhard Gotzhein[1] Daniel Schneider[2]

[1] Computer Science Department, University of Kaiserslautern
Erwin-Schrödinger-Str., D-67663, Kaiserslautern, Germany
{webel, gotzhein}@informatik.uni-kl.de

[2] Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1, D-67663 Kaiserslautern, Germany
Daniel.Schneider@iese.fraunhofer.de

Contents

Abstract

The provision of *network Quality-of-Service* (network QoS) in wireless (ad-hoc) networks is a major challenge in the development of future communication systems. Before designing and implementing these systems, the network QoS requirements are to be specified. Since QoS functionalities are integrated across layers and hence QoS specifications exist on different system layers, a QoS mapping technique is needed to translate the specifications into each other. In this paper, we formalize the relationship between layers. Based on a comprehensive and holistic formalization of network QoS requirements, we define two kinds of QoS mappings. QoS domain mappings associate QoS domains of two abstraction levels. QoS scalability mappings associate utility and cost functions of two abstraction levels. We illustrate our approach by examples from the case study *Wireless Video Transmission.*

1 Introduction

One of the major challenges in wireless (ad-hoc) networks is the provision of *network quality of service* (network QoS), *i.e.* the quality of service provided by the underlying communication system. The need for network QoS arises from the fact that, for state-of-the-art distributed user applications, it is essential, to offer their functionality with a certain degree of quality, which requires suitable communication mechanisms.

State-of-the-art wireless distributed communication systems must offer proactive and intelligent behaviour in order to cope with varying channel quality and connectivity. In other words, wireless communication systems must facilitate changes at runtime, and these changes are to be performed according to an effective reasoning about user, environment, and system context. The realization of such adaptive behaviour can in fact be seen as one of the technological key challenges in the development of wireless communication systems supporting *network quality of service.*

One of the main drivers of adaptive behaviour is the need to maintain specific non-functional properties, *i.e.* a specific level of QoS (Quality of Service) for the provided services. Especially in the wireless domain, where resources (like bandwidth, energy, processing power, and memory) are inherently scarce and subject to frequent change, the systems need to manage their resources in a QoS-aware way. To this end, to form a basis for corresponding adaptation mechanisms to work on, a major prerequisite is an explicit specification of QoS offers and correlated needs. Moreover, QoS as an inherently cross-cutting concern has to be considered from end-to-end and from user layer down to the hardware layer. The QoS specifications hence reside on different layers of abstraction and need to be *mapped* on each other.

In this paper, we we formalize the mapping of different QoS specifications into each other, i.e. the relationship between layers. Based on a comprehensive and holistic formalization of network QoS requirements [3], we define two kinds of QoS mappings. QoS domain mappings associate QoS domains of two abstraction levels. QoS scalability mappings associate utility and cost functions of two abstraction levels.

The remaining part of the paper is organized as follows: In Section 2, we present our case study *Wireless Video Transmission.* Sections 3 sums up our previous work. Section 4 describes the different abstraction levels in communication systems supporting QoS and the mappings between these levels. Last, Section 5 presents conclusions and future work.

2 Case Study *Wireless Video Transmission*

In the following sections, we will use our case study *Wireless Video Transmission* [2] for illustration purposes. It consists of one video data source and one video projection facility, interconnected via a wireless medium. There are three parameters to adjust the video source: video resolution, JPEG compression in percent as a quality factor, and frame rate. The need for adaptive QoS in a wireless environment arises from the heavy network load caused by video transmission. While in wired networks, the available bandwidth is usually sufficient, video transmission can absorb almost all available communication resources in unstable wireless ad-hoc networks, depending on the chosen video frame rate, image quality and resolution. Without a specification of the needed QoS and further mechanisms operating on the specification, this resource consumption may lead to a situation where the video transmission tends to congest the medium and thus prevents all wireless communication.

3 Formalization of Network Quality of Service

In previuos work [3], we have introduced a formalization and specification of network quality of service. This formalization forms the basis for QoS mapping. In this Section we briefly sum up the results from [3].

The need for *formalization* of network quality of service arises from the fact that a *precise* description of network QoS between service user and service provider is needed to police, control, and maintain the data flow a user emits to the communication system. Further on, the mechanisms realizing these functionalities need a *precise* and *well-defined* description of QoS. These mechanisms are typically integrated across layers, and therefore, more than one viewpoint on the required network QoS is needed. So another reason for formalization is the support for a well-defined translation of the specification between the different viewpoints on QoS, called *QoS mapping*. Formalization of network QoS is done by firstly identifying the *QoS domain*, and secondly by describing the *QoS Scalability*.

The *QoS domain* captures the QoS characteristics of a class of data flows, i.e. performance, reliability, and guarantee:

Definition 1 (QoS Domain) *The QoS domain Q is defined as $Q = P \times R \times G$, where P is the performance domain, R is the reliability domain, and G is the guarantee domain. $q = (p, r, g)$ denotes an element of Q, called QoS value.*

QoS performance describes efficiency aspects characterizing the required amount of resources and the timeliness of the service. Therefore, QoS parameters for the requested and provided QoS have to be identified. The relevant aspects are included in the *QoS performance domain* P, which we formalize as follows:

Definition 2 (QoS Performance) *A QoS performance domain P is defined as $P = P_1 \times \ldots \times P_n = \prod_{i=1}^{n} P_i$, where $P_1, \ldots, P_n$ are performance subdomains.*

QoS reliability describes the *safety-of-operation* aspects characterizing the fault behaviour (e.g., loss rate and distribution, corruption rate and distribution, error burstiness). It can significantly impact the overall throughput and functionality on lower system layers, since it requires redundancy (e.g., retransmission, forward error control). The relevant aspects are included in the *QoS reliability domain* R:

Definition 3 (QoS Reliability) *The QoS reliability domain R is defined as $R = Loss \times Period \times Burstiness \times Corruption$, with $Loss = \mathbb{N}_0$, $Period = \mathbb{R}_+$, $Burstiness = \mathbb{R}_+$, and $Corruption = \{r \in \mathbb{R} \mid 0 \leq r < 100\}$.*

QoS guarantee describes the degree of commitment characterizing the binding character of the service. Four degrees of commitment are distinguished: *Best-effort*, *Deterministic*, *Statistical*, and *Enhanced Best-Effort*. QoS guarantee is formalized by the *QoS guarantee* domain:

Definition 4 (QoS Guarantee) *The domain of QoS guarantee G is defined as $G = DoC \times Stat \times Prio$, where $Stat = \{p \in \mathbb{R} | 0 < p \leq 1\}$, $Prio = \mathbb{N}$, and $DoC = \{bestEffort, enhancedBestEffort, statistical, deterministic\}$.*

Varying communication resources require adaptive mechanisms to avoid network overload, and to scale the application service. The *QoS scalability S* describes the control aspects characterizing the scope for a dynamic adaptation of the QoS aspects of a data flow (described by a QoS domain) to a certain granted network quality of service:

Definition 5 (QoS Scalability) *Let Q be a QoS domain. The domain of QoS scalability S is defined as $S = Util \times Cost \times Up \times Down$, where $Util = \{u \mid u : Q \to [0, 1]\}$, $Cost = \{c \mid c : Q \to \mathbb{R}_+\}$, and $Up, Down \in \{x \in \mathbb{R}_+ \mid 0 \leq x \leq 1\}$.*

The elements of *Util* and *Cost* are called *utility functions* and *cost functions*, respectively. A utility function determines the usefulness of QoS values, a cost function c expresses the amount of needed resources, associating higher costs with scarcer resources.

QoS values with the same utility ($\sim_u$) are assigned to the same so-called u-equivalence class of Q:

$$[x]_u = \{q \in Q \mid q \sim_u x\} \tag{3.1}$$

A *QoS requirements specification* defines the set of valid QoS values and a QoS scalability value:

Definition 6 (QoS Requirements Specification) *Let Q be a QoS domain and S be a QoS scalability domain. A QoS requirements specification qosReq is defined as a triple (q_{min}, q_{opt}, s), where $q_{min},\ q_{opt} \in Q$ and $s \in S$.*

The QoS values q_{min} and q_{opt} specify a set $Q^{'} \subseteq Q$ of valid QoS values. To obtain this $Q^{'}$, a preorder $\lesssim_u$ induced by the utility function is applied:

$$Q^{'} = \{q \in Q \mid q_{min} \lesssim_u q \lesssim_u q_{opt}\} \tag{3.2}$$

We can stepwise reduce the QoS domain Q. Therefore, we define the *reduced* QoS domain Q^u by selecting the best element of each u-equivalence class of Q regarding c. Let m be the cardinality of $Q/{\sim_u}$, the quotient set of Q w.r.t. $\sim_u$, and let $[x]_u^i$ denote the ith element of $Q/{\sim_u}$ regarding $\lesssim_u$ (ith u-equivalence class). Then,

$$Q^u = \{q_1, \ldots, q_m\} \cap Q^{'}, \quad q_i = q \in [x]_u^i \mid \forall y \in [x]_u^i \cdot q \lesssim_c y, \quad 1 \leq i \leq m \tag{3.3}$$

A further reduction induces a derived QoS domain $Q^{u,c}$, discarding QoS values with higher cost, but less utility:

$$Q^{u,c} = \{q \in Q^u \mid \forall y \in Q^u \, . \, c(q) > c(y) \Rightarrow u(q) > u(y)\} \tag{3.4}$$

4 QoS Mappings along the Abstraction Levels

Before we elaborate on the concept of QoS mapping, we firstly present our QoS abstraction levels.

4.1 QoS Abstraction Levels

The various QoS mechanisms realizing the QoS specification operate on different system layers, each with its own viewpoint describing the data flow traversing the (communication) system. We call these different viewpoints *QoS Abstraction Levels* and distinguish between four of them:

- *User:* From the user point of view, the needed QoS is usually described in terms of application scenarios. The reason for this is that a user is normally not interested in the concrete details and parameters of a QoS communication, but on the consequences for him.

- *Application:* On application level, the quality of service is specified in terms of application parameters describing the user data flow. The user quality of service is translated into more concrete parameters. For example, a video transmission is described by a video frame rate, resolution or image quality. This is done in a hardware and platform independent way.

- *Communication:* The underlying transport system, the communication network, provides a further viewpoint on quality of service. On this abstraction level, transmission units, transmission periods, packet delay or packet jitter describe the needed network QoS to fulfill the QoS requirements in an application and hardware independent, but platform specific manner.

- *Resource:* Finally, for a QoS communication to be realized on a specific platform and base technology, the parameters of the communication abstraction layer have to be refined to support concrete resources such as bandwidth, energy, cpu cycles or memory.

On each abstraction level, a *QoS requirements specification* describes the needed and granted QoS in a formal manner. A *QoS mapping* translates the different specifications into each other.

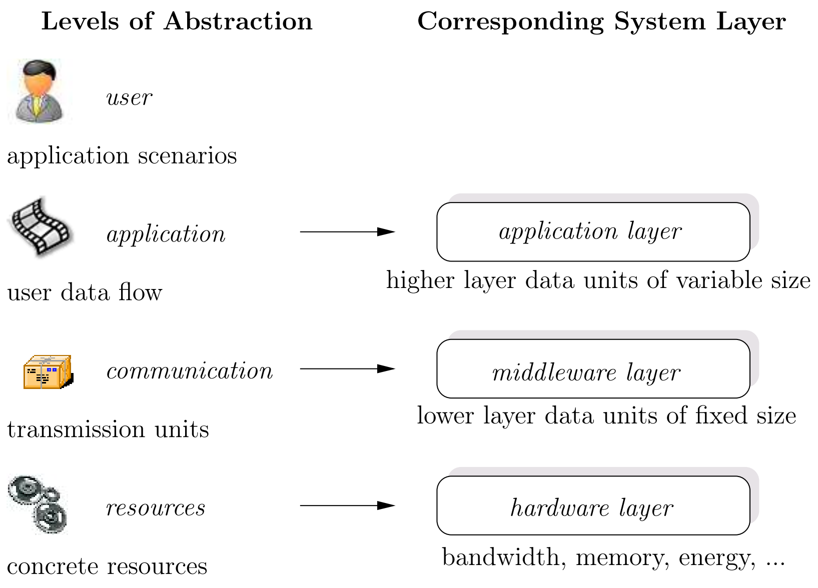

Figure 4.1: Different levels of abstraction and corresponding system layers

In Figure 4.1 abstraction levels are associated with corresponding system layers. The user point of view is not mapped to a specific system layer, as it is not part of the system.

In the *application layer*, most of the medium- and long-term QoS functionalities can be found. Here, the user data flow is application dependently processed or manipulated. The units on this layer are typically higher layer protocol data units (PDUs) of variable size. The application layer is usually tailored to one or more applications. Within this layer, there is no knowledge of the underlying communication system. Thus, the QoS specification on this layer is application-specific but communication system- and hardware-independent and the level of abstraction corresponds to the application abstraction level.

The communication abstraction level corresponds to the *middleware layer*. Here, the transport units are lower layer PDUs of a (fixed) maximum size. Different user data flows are multiplexed together to form only one data flow to be transmitted via the network. On this layer, short-term QoS functionalities control and manipulate the network data, *e.g.* traffic shaping and packet scheduling. An exact knowledge of the platform, *i.e.* the operating system and the base technology, or the current network state, *e.g.* topology or channel conditions, is not required. So the QoS specification on the middleware layer is platform- and application-independent.

The most fine-grained level of abstraction, the resource point of view, is mapped to the *hardware layer*. The specification of QoS is directly translated into concrete resources such as bandwidth, energy, memory and so on. Typical QoS functionalities are medium

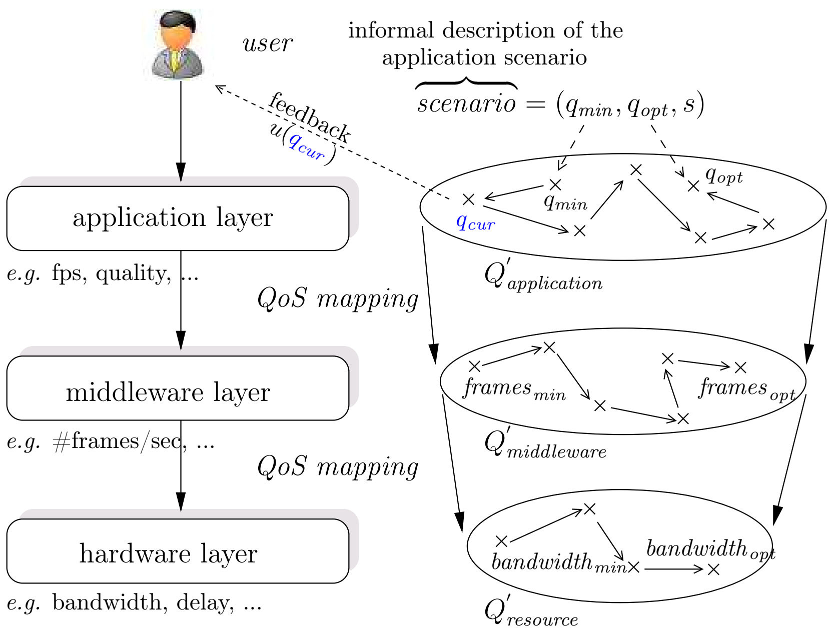

Figure 4.2: Mapping of two QoS requirements specifications

access and resource reservation. For this reason, the QoS specification on this layer is highly hardware dependent. Usually it is not possible to directly specify the needed QoS on this system layer with an appropriate level of detail, since too many factors and interrelationships between the hardware parameters have to be considered. For instance it is not possible to manually specify the needed energy for a given operation, since it depends on the cpu cycles, the memory allocation or a resulting transmission of data.

4.2 QoS Domain Mapping

To relate QoS requirements of different abstraction levels, QoS mappings are needed. In this section, we introduce *QoS domain mappings* between a higher layer QoS domain Q_h and a lower layer domain Q_l.

This is illustrated in Figure 4.2. A service user describes the application scenario in an informal way, *e.g.* *surveillance* or *panorama*. Based on this informal description, a detailed QoS requirements description on application layer has to be derived. This cannot be done automatically, since it is not possible to automatically generate a concrete specification out of an informal description. So the first mapping between user and application layer is explicitly given by the tight coupling between these two layers. Based on this first requirements specification, a subset $Q'_{application}$ (cf. (3.2)) is formed on application layer. The different communication layers use different parameters to describe the user data flow and hence the available network QoS. On application layers, *e.g.* a video data flow is described in terms of video frames per second and a quality factor.

This description has to be translated to a corresponding specification on middleware layer, where the parameters are *e.g.* frames per second. The reason for that is, that mechanisms on this layer like segmentation and composite or error detection and correction operate on a user data independent representation of the data flow. Finally, the specification on middleware layer has to be mapped to a specification on hardware layer, taking into account concrete resources like bandwidth, delay, or even energy consumption. The QoS mapping is obviously nontrivial. The user feedback of the currently granted QoS is given via the utility function.

Definition 7 (QoS Domain Mapping) *Let Q_h, Q_l be QoS domains on different system layers. A QoS domain mapping $dm : Q_h \rightarrow Q_l$ is a function from a (higher layer) QoS domain Q_h to a (lower layer) QoS domain Q_l. The domain mapping dm may be defined using auxiliary functions as follows:*

$$\begin{aligned} dm_P &: Q_h \rightarrow P_l \text{ (performance mapping)} \\ dm_R &: Q_h \rightarrow R_l \text{ (reliability mapping)} \\ dm_G &: Q_h \rightarrow G_l \text{ (guarantee mapping)} \end{aligned}$$

In general, the QoS mappings are neither injective nor surjective. That means, that two different QoS values $q_1, q_2 \in Q_h$ could be mapped to the same $q_l \in Q_l$ and that the values of m do not span the whole codomain Q_l Figure 4.3 illustrates the problem. Given four elements of a reduced QoS domain $Q^{'}$ and an utility functions u describing an order on $Q^{'}$. The first QoS mapping dm_1 is injective, *i.e.* the mapping function is order-preserving under a new order $\leq_n$ with $\forall\, q_1, q_2 \in Q_l : q_1 \leq_n q_2 \Leftrightarrow u(dm_1^{-1}(q_1)) \leq_u u(dm_1^{-1}(q_2))$. But often, more than one higher layer tuple is mapped to the same lower layer tuple (dm_2). In this case, the mapping function is non order-preserving under u, since, on the one hand, no inverse mapping function dm_2^{-1} exists, and, on the other hand, $(1, 3)$ cannot be simply compared to $(3, 3)$. Also there exists tupels (x, y) with $dm(q^{'}) \neq (x, y)$. For these reasons, the mapping of the scalability requirements specification, especially the utility function, is nontrivial.

In the following, we elaborate on the three subfunctions in detail.

4.2.1 QoS Performance Mapping

The QoS performance mapping dm_P translates the performance parameters into each other. The performance parameters are system layer and hardware dependent, i.e. parameters like the maximum transfer unit (MTU), the path MTU, or the frame format have to be considered.

Definition 8 (QoS Performance Mapping) *Let P_h, P_l be performance domains on different system layers. A QoS performance mapping $dm_P : P_h \rightarrow P_l$ is a function translating performance values $p_h \in P_h$ into new values $p_l \in P_l = P_{l_1} \times \cdots \times P_{l_n}$. To define dm_P, auxiliary functions $dm_{P_i}(p_h) = p_{l_i},\ \forall\, i \leq l_n$ can be used.*

$m_1 = (a, b+c)$

$$\begin{tikzcd}
(1,1,2) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_1"] & (1,3,2) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_1"] & (2,3,1) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_1"] & (3,1,0) \arrow[d, "dm_1"] \\
(1,3) \arrow[r, phantom, "\leq_n"] & (1,5) \arrow[r, phantom, "\leq_n"] & (2,4) \arrow[r, phantom, "\leq_n"] & (3,1)
\end{tikzcd}$$

$m_2 = (b, a+c))$

$$\begin{tikzcd}
(1,1,2) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_2"] & (1,3,2) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_2"] & (2,3,1) \arrow[r, phantom, "\leq_u"] \arrow[d, "dm_2"] & (3,1,0) \arrow[d, "dm_2"] \\
(1,3) \arrow[r, phantom, "\leq_n"] & (3,3) \arrow[r, phantom, "\leq_n"] & (3,3) \arrow[r, phantom, "\leq_n"] & (1,3) \ \lightning
\end{tikzcd}$$

Figure 4.3: Example of an order-preserving (m_1) and a non order-preserving mapping function (m_2)

Referring to our case study, a performance mapping from the application layer performance parameters P_{video} to the underlying middleware layer with $P_{mw} = \#Frames \times Period$ is done in the following way:

$$dm_P : P_{video} \rightarrow P_{mw}$$
$$dm_P((res_x, res_y), fps, quality) = (\#frames, period), \text{ with}$$
$$dm_{P_1}((res_x, res_y), fps, quality) = \#frames = \left\lceil \frac{(160 \cdot quality + 3000) \cdot (res_x - 160)/160}{user\ \ bytes\ \ per\ \ frame} \right\rceil$$
$$dm_{P_2}((res_x, res_y), fps, quality) = period = \frac{1}{fps}$$

The middleware protocol data unit *frame* contains a maximum amount of 1420 user bytes. Given quality and resolution, the number of transmission frames per picture frame can be determined. *Period* specifies a deadline interval of the periodic transmission task.

4.2.2 QoS Reliabiliy Mapping

Higher layer transmission units (e.g., picture frames) can be larger than lower layer units and therefore have to be fragmented and reassembled. This, however, complicates the definition of the QoS reliability mapping (see [1]). To illustrate this, consider the example in Fig. 4.4. On application layer, the variable-size picture frames are fragmented into maximum-size middleware packets. On middleware layer, we assume a loss ratio of 30%. The loss can be caused by packet loss, corrupted, dropped, or late-delivered PDUs. Further, we assume that a loss of even one lower layer packet results in the loss of the entire picture frame. In Figure 4.4.a, the loss ratio results in a picture frame loss of 33%. If the loss is uniformly distributed (as shown in Fig. 4.4.b), the same ratio leads to a loss on application layer of 100%.

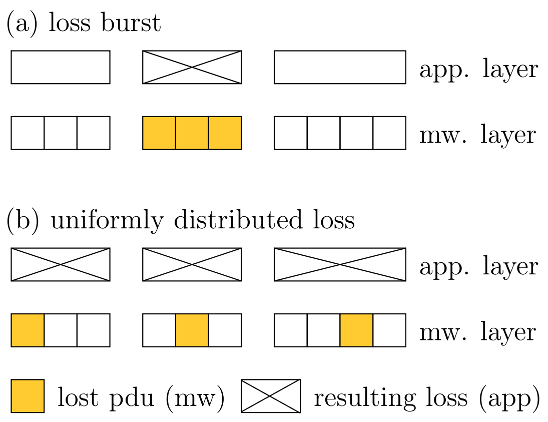

Figure 4.4: Upper layer PDUs vs. lower layer PDUs

Notice that a simple description of the lower layer loss or corruption probability is not sufficient for deriving the expected upper layer reliability parameters. Moreover, uniformly distributed losses may be more adverse than bursty losses. To define the QoS reliability mapping, a segmentation model of the user data is needed. In our case study, this model would introduce probability distributions of picture frame sizes and resulting probability mass functions of the number of needed middleware packets. Further, an error model characterizing the loss and/or corruption process is needed. This error model strongly depends on the chosen base technology. The definition of segmentation and error model are out of the scope of this paper. A treatment of these aspects can be found in [1].

4.2.3 QoS Guarantee Mapping

The function dm_G maps the guarantees specified on one system layer to corresponding guarantees on another. Ideally the guarantees should stay the same during a mapping process. But in exceptional cases, e.g., if the underlying base technology does not support required degree of commitment, an upgrade is permitted. For example, a mapping from *statistical* to *deterministic* guarantees is always feasible, whereas a mapping vice versa could result in a violation of the traffic contract.

4.3 Scalability Mapping

QoS scalability describes the control aspects characterizing the scope for dynamic adaptation of QoS parameters. To apply scaling on different levels of abstraction, a QoS scalability mapping is needed. For consistency, this mapping has to ensure that the utility of QoS values of different abstraction levels that are related by the QoS domain mapping dm is the same. To enforce this consistency condition, we will now define a scalability mapping such that, given a utility function u_h, yields the corresponding utility function u_l. Next, we will introduce a *cost function* that associates costs with QoS

$$\begin{tikzcd}
\textit{higher layer:} & S_h & = & Util_h \arrow[d, "sm_{Util}"] & \times & Cost_h & \times & Up_h \arrow[d, "sm_{Up}"] & \times & Down_h \arrow[d, "sm_{Down}"] \\
\textit{lower layer:} & S_l & = & Util_l & \times & Cost_l \arrow[u, "sm_{Cost}"'] & \times & Up_l & \times & Down_l
\end{tikzcd}$$

Figure 4.5: Scalability mapping

values. Based on this cost function, we will finally arrive at a reduced set of QoS values characterizing the actual scope for dynamic adaptation.

In the following definition, let $Q_l^* = \{q_l \in Q_l \mid \exists\, q_h \in Q_h \,.\, dm(q_h) = q_l\}$ denote the set of mapped QoS values, and let $[q_h]_{dm} = \{x \in Q_h \mid x \sim_{dm} q_h\}$, $\sim_{dm} = \{(q_h, q_h^{'}) \in Q_h \times Q_h \mid dm(q_h) = dm(q_h^{'})\}$, denote the equivalence classes containing those QoS values q_h that are mapped to the same q_l.

Definition 9 (QoS Scalability Mapping) *Let S_h, S_l be scalability domains on different system layers. A QoS scalability mapping is a set of four mapping functions sm_{Util}, sm_{Cost}, sm_{Up} and sm_{Down}, translating the different scalability domains into each other (see Fig. 4.5):*

$$\begin{aligned}
sm_{Util} &: Util_h \to Util_l;\ \forall u_h \in Util_h \,.\, \forall q_l \in Q_l^* \,.\, u_l(q_l) =_{DF} u_h(q_h) \mid \\
&\qquad\qquad q_h \in Q_h \wedge dm(q_h) = q_l \wedge \forall x \in [q_h]_{dm} \,.\, u_h(q_h) \geq u_h(x) \\
sm_{Cost} &: Cost_l \to Cost_h;\ \forall c_l \in Cost_l \,.\, \forall q_h \in Q_h \,.\, c_h(q_h) =_{DF} c_l(dm(q_h)) \\
sm_{Up} &: Down_h \to Up_l; \forall x \in Up_h \,.\, sm_{Up}(x) =_{DF} x \\
sm_{Down} &: Down_h \to Down_l; \forall x \in Down_h \,.\, sm_{Down}(x) =_{DF} x
\end{aligned}$$

Some explanations are in order. Let $q_h, q_h^{'} \in Q_h,\ u_h(q_h) > u_h(q_h^{'}),\ dm(q_h) = dm(q_h^{'})$. In other words, although the utility of q_h is higher than that of $q_h^{'}$, they consume the same amount of resources $q_l = dm(q_h)$. In this case, the utility of q_l is chosen as $u_l(q_l) =_{\mathrm{DF}} u_h(q_h)$, i.e. the better value. This means, that when the resources q_l are available, they are exploited as best as possible. This idea is generalized in the definition of the mapping function sm_{Util}, where to each value of $q_l \in Q_l^*$, the maximum utility of all corresponding values $q_h \in Q_h$ is assigned. Note that costs are mapped from lower to higher system layer and that the thresholds for upscaling and downscaling remain unmodified by the QoS scalability mapping.

With the QoS mappings dm, sm and the reduced QoS domain (see (3.4)), it is possible to define a *scaling function* to be used in system design and implementation. The scaling function $scal_{u,c_l} : Q_l \to Q^{u,c_h} \cup \{0\}$ forms the main part of our scalability model and therefore maps a lower layer QoS values describing the currently granted network QoS to a higher layer *cost-optimal* QoS value. The function selects the best possible, i.e. the *optimum* QoS value $q \in Q^{u,c_h}$ regarding the utility function u in compliance with the currently granted QoS resources $q_{granted} \in Q_l$, if such an element exists, otherwise 0. For this reason, the cost function c_l has to be mapped to a corresponding higher layer cost function c_h in order to properly reduce Q. The scaling function is defined as follows:

$$scal_{u,c_l}(q_{granted}) = \max_u\{q \in Q^{u,sm_{Cost}(c_l)} \mid c_l(dm(q)) \leq c_l(q_{granted})\} \tag{4.1}$$

whereas the maximum operator max_f for a given set X defines x as an f-maximal element of X iff $x \in X$ and $\forall y \in X : (f(x) \leq f(y) \Rightarrow x = y)$, short $\max_f\{X\}$. The maximum of an empty set is defined as zero, i.e. $\max_f \varnothing = 0$.

5 Conclusion and Future Work

In this paper, we have presented two kinds of QoS mappings, called QoS domain mappings and QoS scalability mappings. These mappings are based on an holistic, comprehensive formalization of network QoS requirements. The QoS domain mapping relates QoS domains on different system layers using three auxiliary functions, a performance mapping, a reliability mapping, and a guarantee mapping. The Qos scalability mapping translates QoS scalability domains into each other, across layers.

All mappings so far are based on mathematics, since the formalization of network QoS requirements is based on it. For better usability, we intend to define a formal QoS requirement specification language, with intuitive keywords and structuring capabilities. This language should be powerful enough to host the concepts and criteria we have introduced in this paper and in por prvious work. Also, the language should be supported by tools that can, for instance, construct QoS mappings as far as they have been defined in this work.

Acknowledgment

The work presented in this paper was (partially) carried out in the BelAmI (Bilateral German-Hungarian Research Collaboration on Ambient Intelligence Systems) project, funded by German Federal Ministry of Education and Research (BMBF), Fraunhofer-Gesellschaft and the Ministry for Science, Education, Research and Culture (MWWFK) of Rheinland-Pfalz.

Bibliography

[1] L. A. DaSilva. QoS Mapping Along the Protocol Stack: Discussion and Preliminary Results. In *Proceedings of IEEE International Conference on Communications (ICC'00)*, volume 2, pages 713–717, New Orleans, LA, June 18-22, 2000.

[2] C. Webel, I. Fliege, A. Geraldy, R. Gotzhein, M. Krämer, and T. Kuhn. Cross-Layer Integration in Ad-Hoc Networks with Enhanced Best-Effort Quality-of-Service Guarantees. In *Proceedings of World Telecommunications Congress (WTC 2006)*, Budapest, Hungary, 2006.

[3] C. Webel, R. Gotzhein, and D. Schneider. Formalization of quality-of-service requirements for wireless networks. Technical Report 356/07, University of Kaiserslautern, Kaiserslautern, Germany, 2007.