Generic Function Schema for Operations on Multiple Network QoS Parameters

Mark Yampolskiy[1,2,3], Wolfgang Hommel[2,3], David Schmitz[1,2,3], Matthias K. Hamm[3]
[1]*German Research Network (DFN),* [2]*Leibniz Supercomputing Centre (LRZ),* [3]*Munich Network Management (MNM) Team*
myy@dfn.de, hommel@lrz.de, schmitz@lrz.de, hamm@mnm-team.org

***Abstract*—Graphs are often used to model interconnected topological objects with different connection properties. Path finding in a weighted graph belongs to the classical problems of graph theory. Whereas the addition of the edges' weights as an aggregation and the interpretation of a smaller resulting sum as the preferable path works very well in applications like path computations, e.g., for road maps, it is not always applicable to those connections in computer networks that need to fulfill multiple independent Quality of Service (QoS) criteria in parallel. Until now, usually a special – and often manual – solution has been implemented for each new service with different QoS parameters. As the development of novel customer-faced network services often relies on different connection properties and their combinations, a generic treatment of QoS parameters becomes a critical factor for rapid development and network service rollout. In this paper, we present our proposal for treating multiple independent QoS parameters in a similarly fashioned way. Our work is aimed to foster routing algorithms that are considering multiple connection properties and corresponding constraints at the same time.**

***Keywords*-graph theory, multi-weighted graphs, QoS, QoS aggregation, QoS comparison.**

I. Introduction

Obviously, network connections are meanwhile broadly used as a basis or an integral part of the services realized upon them. Examples can be found in areas like internet-telephony, video-conferencing and video-on-demand, connectivity for GRID cooperation, etc. Common to all these examples is that the overall service quality directly depends on the combination of multiple Quality of Service (QoS) parameters of the underlying network connections. For example, regarding telephony those QoS parameters are bandwidth, low delay, and low jitter; video on demand – depending on the actual service model – might be more jitter-tolerant but instead requires much higher bandwidth. Network connections dedicated to the distribution of experimental data in the Large Hadron Collider (LHC) project [1] should provide dedicated bandwidth, high availability, and low maintenance time. In order to cope with a larger variety of customer and service requirements, miscellaneous QoS parameters as well as their combination should be considered during path computation (routing) for the connection setup.

Graphs are often used to model interconnected topological objects with different connection properties. Path finding in a weighted graph belongs to the classical problems of graph theory. Whereas the addition of edge-weights as an aggregation functions and the treatment of smaller resulting sum as the best path works very well in applications like path computations, e.g., for road maps, it is not applicable to connections in computer networks with multiple independent QoS parameters in general. Already the two QoS parameters considered most often, i.e., bandwidth and delay, show significant differences. Whereas the usual parameter treatment is applicable to delay, for bandwidth different functions are needed: the aggregation function is minimum and larger values are preferred to smaller results. The adequate QoS aggregation functions are significantly more complex than sum-of or minimum-of if other QoS parameters, e.g., reliability and availability, or aspects, which are relevant for service instance management, like maintenance windows for multi-domain connections, have to be considered.

Until now, in practice usually a special solution has been implemented for a service that required new or different QoS parameters. Nowadays the time for the development of a new service with customer-specific QoS parameters is becoming a crucial success factor. Therefore, a general treatment of QoS parameters is absolutely critical in order to ensure sufficiently fast adaptability and extensibility of already existing and new services. In the first place, an efficient way to distinguish between different QoS parameters is needed. Furthermore, a standardized general treatment for the aggregation of and the comparison between values of a particular QoS parameter is indispensible. As different customer-faced services might depend on different subsets of those QoS parameters, the efficient support of customer-relevant combinations of arbitrary QoS parameters is needed.

The remainder of this paper is structured as follows: In Section II the related work, which has influenced our solution is presented. The proposal for a generic treatment of different QoS parameters is described in Section III. It includes the distinction between the various QoS parameters and the definition of functions necessary during the routing process. Furthermore, we generalize the way to handle multiple QoS parameters simultaneously. In Section IV we extend our proposal in order to support value ranges as edge weights. In Section V we present how our proposal can be applied to the definition of optimized routing algorithms. A short outlook to our future work concludes this paper.

II. State of the Art and Road Map

Typically, graphs are considered that have single fixed values associated with their edges as weights. This repre-

sentation is usually used for finding a path or a shortest path between two endpoints in a graph. However, such graphs only conditionally reflect all specifics of computer networks (see Figure 1). For instance, due to the support of different quality classes of the used network infrastructure, the property value supported by a single connection can vary in a broad range. In order to process value ranges, which are supported, e.g., by the information model described in [2], a transformation to so called *multigraphs* is possible. In the case of multigraphs, nodes may be directly connected by one or more edges. Even in a simple case with weights for a single property, such transformations can significantly increase the graph complexity. If multiple connection properties with value ranges have to be considered at the same time, the complexity increases start to be even more drastically.

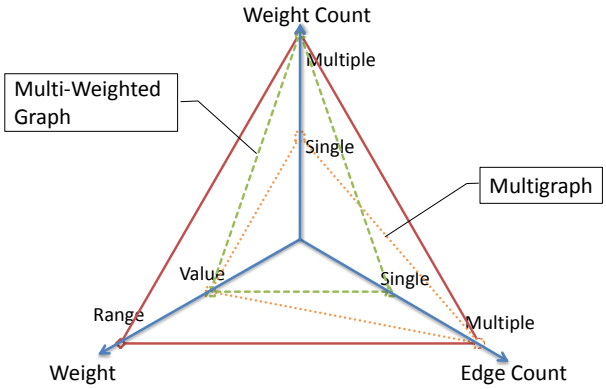

Figure 1. Graph properties, classification

Graphs that support multiple properties at the same time are known as *multi-weighted graphs*. Such graphs are hardly investigated yet. In [7], a very good overview about the state of the art is given, and various problems and solution ways are investigated. In summary, path finding in multi-weighted graphs is in general a $NP-complete$ problem. As for path finding in multi-weighted graphs the Bellman's optimality principle [8] is not fulfilled, broadly used routing algorithms that require this principle, e.g., Dijkstra's algorithm, cannot be used. Among other aspects, also the handling of multiple properties at the same time has to be solved. Currently, the common understanding is to describe multiple properties as value vectors. This allows the use of vector-addition as property aggregation operation. For a comparison of weight vectors, the concept of *non-dominance* has been established, as it is described and used in [4]. A vector A is non-dominant to vector B only if all of its weight elements, i.e., property values, are smaller or equal to the corresponding elements of vector B.

Considering operations for both single- and multi-weighted graphs, addition is used as an aggregation function and the smaller value is treated as the better one. Even if limitations of these operations w. r. t. the application to computer networks are long known to the research community, until now only workarounds have been proposed. For instance, in [9] an addition of $log(weight)$ is proposed, if the true aggregation function for $weights$ is multiplicative.

As a summary of the above discussed missing aspects, in order to enable operations on graphs describing computer networks with arbitrary supported properties, the following extensions have to be implemented:

- Support for arbitrary functions for aggregation and comparison of weights of a single connection property.
- Operations on bundles of properties, which could be used in multi-weighted graphs.
- Improved handling of value ranges.

Solutions for the first two aspects will be described in Section III. A proposal for handling of value ranges during path finding will be presented in Section IV-A.

Besides these purely technical aspects, also organizational specifics have to be considered. The so called policy-based routing, which is used for inter-domain routing, in the first place takes into account not only technical aspects, but rather provider-specific interests. Along with very restrictive information and management policies, which are out of the scope of this paper, SPs are generally interested in the reduction of resources used for a service delivery. A corresponding proposal will be given in Section IV-B.

III. Operations on Connection Properties and their Generalization

In this section we present our solutions for function generalization regarding both single properties and property bundles. The important extension for the treatment of value ranges is described in Section IV.

A. Functions for operations on a single property

During the path finding, the properties of the edges have to be aggregated. Typically, simple arithmetical addition is used as an aggregation function. As discussed in Section I, this is not necessarily the case for any QoS parameter. Furthermore, as discussed in [2], in the case of the inter-domain connections each Service Provider (SP) might have access only to its own infrastructure, which might not be sufficient to determine all the relevant connection properties. In this case also the aggregation of the partial views of involved SPs at the same inter-domain connection is needed. The calculation of QoS properties of the inter-domain link from two partial views is not necessarily identical to the aggregation of two physical connections of the same type and length. For instance, when describing a connection with the property "delay", not only the delay caused by the network cable should be considered, but also the delay caused by the active and passive network components used by each single SP; obviously, it typically varies between SPs.

If customer-specific end-to-end quality-of-service constraints shall be met, the value of the already found (partial)

route has to be compared to these constraints during the path finding process. For path optimization it is also necessary to compare the values of two alternatives in order to choose the better one. In opposite to the case classically treated in graph theory, the meaning of what is "better" might vary between different QoS parameters. Regarding the examples mentioned above, for bandwidth a bigger value can be considered as a better one, however for delay a smaller value is the more preferred one.

Consequently, with each supported connection property operations for value aggregation and comparison have to be associated.

B. Associating operations with properties

In IT industry, new technologies and services are evolving very fast. Therefore prior the association of operations with properties, a distinction between existing and upcoming properties is needed. We propose to assign a globally unique ID to each supported property. In order to ensure the global uniqueness of IDs, we propose to use a registration tree. Additionally to the distinction between properties, using a registration tree has another very important advantage. As multiple functions have to be associated with each supported property, it can be realized by the definition of the functions together with the registration of their property-ID (see Figure 2). Additionally this will ensure the identity of functions used among multiple SPs.

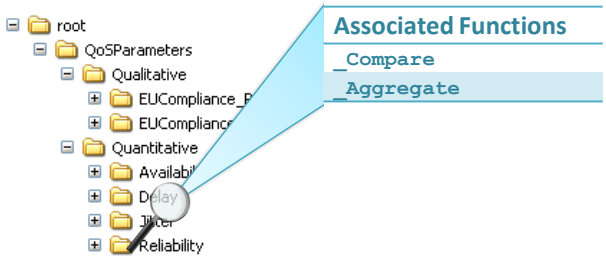

Figure 2. Registration tree example

C. Comparison and aggregation of multiple properties

Based on the previous definition, we introduce an approach for the handling of m different properties with the global unique IDs $\mathrm{ID}_1, \ldots, \mathrm{ID}_m$. In graph theory, it is a common practice to use vectors in order to describe multiple weights associated with a single edge or a path in general. For any path in a graph with m properties, the weight can be specified as $\overrightarrow{U} ::= (u_1, \ldots, u_m) \in \mathbb{R}^m$. In this definition, u_j is the weight of the $j^{\underline{th}}$ property with ID_j. The order of properties in the weight vector can be arbitrary, as long as the placement of the properties is identical among all weight vectors. Further, for the edges of a path being enumerated from 1 to n, the weight of an edge with index i will be referred to as follows $\overrightarrow{W^i} ::= (w_1^i, \ldots, w_m^i) \in \mathbb{R}^m$.

In order to calculate the weight vector $\overrightarrow{P}$ of the path consisting of n edges with weights $\overrightarrow{W}^1, \ldots, \overrightarrow{W}^n$, we first introduce an aggregation function for two weight vectors as follows:

$$\overrightarrow{\mathrm{Aggr}}(\overrightarrow{U}, \overrightarrow{V}) ::= (\mathrm{Aggr}_1(u_1, v_1), \ldots, \mathrm{Aggr}_m(u_m, v_m))$$

This definition is based on m aggregation functions for each property. The aggregation functions Aggr_i $(i = 1, \ldots, m)$ are functions associated with the property ID in the registration tree. We assume that all properties are independent of each other, i.e., they can vary without influencing the values of other properties. Furthermore, we assume that the binary operations defined by aggregation functions fulfill associative and commutative laws. Then we inductively define the computation of the whole path weight from weights of involved segments as follows:

$$\overrightarrow{\mathrm{Aggr}}(\overrightarrow{W}^1, ..., \overrightarrow{W}^n) ::= \overrightarrow{\mathrm{Aggr}}(\overrightarrow{\mathrm{Aggr}}(\overrightarrow{W}^1, \overrightarrow{W}^2), ..., \overrightarrow{W}^n)$$

Similar to the aggregation, we define the comparison of property vectors based on the comparison between identical properties. Corresponding to the *non-dominance* concept as it is described in [4], we define that vector $\overrightarrow{U}$ is better than $\overrightarrow{V}$ if and only if all properties in the first vector are better than the corresponding properties of the second vector. In order to depict that property u_i of vector $\overrightarrow{U}$ is better than the corresponding property v_i of vector $\overrightarrow{V}$, we use the symbol "$\prec$". In contrast to the comparison of single values, it is possible that some properties of the first vector are better and some others are worse than of the second vector. This situation should be treated as indefinite. We depict this with symbol "$\neq$". The comparison of two property sets can thus be defined as follows:

$$\overrightarrow{Compare}(\overrightarrow{U}, \overrightarrow{V}) ::= \begin{cases} =, \text{ if } & \forall 1 \le i \le m : u_i = v_i \\ \prec, \text{ if } & \forall 1 \le i \le m : (u_i \prec v_i \\ & \quad \vee\ u_i = v_i) \wedge \\ & \exists 1 \le j \le m : u_j \prec v_j \\ \succ, \text{ if } & \forall 1 \le i \le m : (u_i \succ v_i \\ & \quad \vee\ u_i = v_i) \wedge \\ & \exists 1 \le j \le m : u_j \succ v_j \\ \neq, \text{ if } & \exists 1 \le i \le m : u_i \prec v_i \wedge \\ & \exists 1 \le j \le m : u_j \succ v_j \end{cases}$$

IV. Treatment of Value Ranges

Some important aspects that are typical for computer networks are not directly addressed by classical graph theory. In this section we propose the treatment of value ranges, which can be associated with connection segments (graph edges) instead of multigraphs with multiple alternative connection segments with different fixed values.

A. Path finding with value ranges

Physical network connections usually cannot be realized with a single property set, because properties like bandwidth might vary in a wide range. In the case that an abstracted network description is considered, further connection properties can vary in a wide range. A good example is the variation of achievable delays for a single logical connection, as it can be realized by different physical connections. Consequently, also the property of the whole End-to-End (E2E) path between two endpoints might vary in a wide range. We will refer to the value range of a particular path $path$ as

$$\overrightarrow{\overrightarrow{W}}^{path} = (\overrightarrow{W}^{path}_{min}, \overrightarrow{W}^{path}_{max}) \in \mathbb{R}^m \times \mathbb{R}^m,$$

i.e., the supported value range for the given path can vary from $\overrightarrow{W}^{path}_{\text{min}}$ to $\overrightarrow{W}^{path}_{\text{max}}$.

It is obvious that the path found between two endpoints can only be feasible if the best possible value fulfills the E2E constraints specified by customer (see Figure 3). Therefore we propose to operate with the best values of the available connection segments during the path finding process.

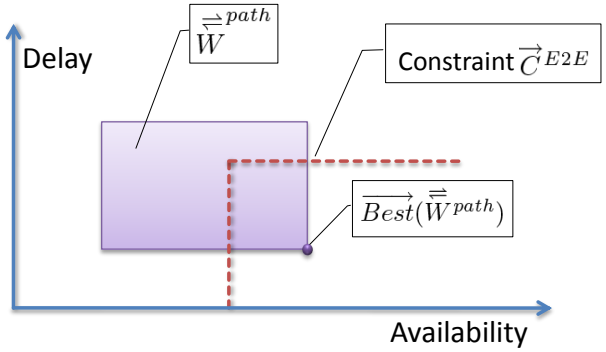

Figure 3. Fulfillment of end-to-end constraints

We assume that all simultaneously considered path properties can vary independent of each other. Under this assumption, we define the selection function $Best$ for the best possible value of a path as follows:

$$\overrightarrow{Best}(\overrightarrow{\overrightarrow{W}}^{path}) = \overrightarrow{Best}(\overrightarrow{W}^{path}_{min}, \overrightarrow{W}^{path}_{max})$$

$$\overrightarrow{Best}(\overrightarrow{U}, \overrightarrow{V}) ::= (Best_1(u_1, v_1), \ldots, Best_m(u_m, v_m))$$

$$Best_i(u_i, v_i) ::= \begin{cases} u_i, & \text{if } u_i \prec v_i \\ v_i, & \text{otherwise} \end{cases}$$
$$\text{for } 1 \leq i \leq m$$

Please note that this definition is applicable not only to a path as a whole, but also to any path segments.

B. Considering service provider interests: Optimization of resource usage

In contrast to customers, the service providers are usually interested in a reduction of resources used for service realization. This means that the requested service quality should not be the best possible one, but rather the one closest to the customer constraints. For paths complying with the E2E constraints, i.e., $\overrightarrow{Best}(\overrightarrow{\overrightarrow{W}}^{path}) \prec \overrightarrow{C}^{E2E}$, we distinguish between three cases as depicted in Figure 4, given the weights of alternative paths A, B and C:

- All worst possible properties of the considered path are worse than the constraints (see "Path A")
- All worst possible properties of the path are better than the constraints (see "Path B")
- The worst possible properties of the path are for some properties worse and for other properties better than constraints (see "Path C")

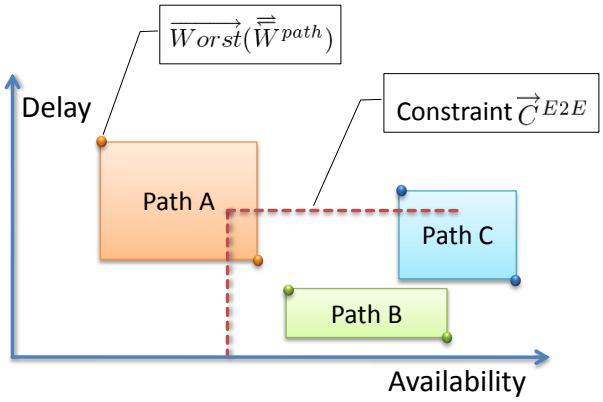

Figure 4. Pathweights of paths complying to constraints

In order to distinguish between these alternatives, the function $Worst$ for the selection of the worst possible value of the found path can be defined as an opposite to $Best$.

In the case equivalent to "Path B", the worst possible value can be requested during the link ordering process. In the two remaining cases, an approximation to the constraint value should be performed. As the properties are independent of each other, such an approximation can be done separately (or even in parallel) for each affected property.

The whole E2E path weight is the sum of the weights of the involved parts. A possible gradation between the maximum and minimum values of connection parts is depicted in Figure 5. The E2E approximation of the path weight for a single property can be done in different ways. It can be seen as a knapsack-like problem with an intention to find a fit most close to the E2E constraint. We argue against this approach, as it may prevent the on-demand adaptation of requested service parts parameters. Instead we favor a "fair split" among all connection parts. For each property i, we propose to use a divide-and-conquer strategy as follows:

1) For each connection part j with a value range between $w^j_{i,min}$ and $w^j_{i,max}$ we compute values $w^j_{i,best}$=Best$_i$ ($w^j_{i,min}$, $w^j_{i,max}$) and $w^j_{i,worst}$=Worst$_i$($w^j_{i,min}$, $w^j_{i,max}$).
2) For each connection part j we compute the realizable value $\left\lfloor \frac{w^j_{i,best}+w^j_{i,worst}}{2} \right\rfloor$.
3) If the computed path value $\sum_{j=1}^{k} \left\lfloor \frac{w^j_{i,best}+w^j_{i,worst}}{2} \right\rfloor$ is equivalent to the E2E constraint for the selected property, the selected values can be used as a result of this optimization.
4) If the computed path value is better than the E2E constraint, the computed values for connection parts should be used in the next step as $w^j_{i,best}$, otherwise as $w^j_{i,worst}$.
5) We propose to limit the number of optimization steps. If the number of maximal optimization steps is reached, the latest $w^j_{i,best}$ for each connection part should be used as an approximation value. If the amount of the maximum optimization steps is not reached yet, this procedure shall be repeated beginning with step (2).

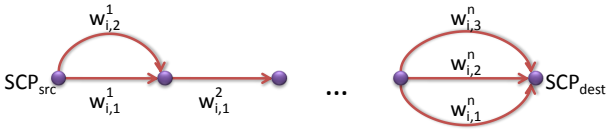

Figure 5. Possible gradation of values for different path segments for property i

Please note that in order to reflect the "better/worthier" comparison instead of "smaller/bigger" one, we define the unary operator "$\lfloor\ \rfloor$" as follows: the result should be the worst realizable value, which is equal or better than the value enclosed in the brackets.

V. Application of Search Problems

In Figure 6, we present a path finding algorithm, which illustrates the usage of our operators. In the pseudo-code, a deep first search strategy is used for finding a path complying with multiple QoS constraints $\overrightarrow{C}^{E2E}$.

The presented algorithm solves the so-called *multi constrained path finding* (MCP) problem. The function requires four parameters. The first two parameters ($nodeCur$ and $nodeDest$) specify nodes in the graph, between which a path has to be found. As the MCP function is called recursively, the $nodeCur$ specifies the end of the intermediately considered path. The weight of the intermediate path is given in the third parameter $\overrightarrow{W}^{path2cur}$. Finally, $\overrightarrow{C}^{E2E}$ are always the E2E-constraints between two endpoints.

In the function, at first it is checked whether the destination node is reached yet. If it is the case, the $BacktracePath$ function is called in order to memorize

```
MCP (nodeCur, nodeDest, W^path2cur, C^E2E)

 if (nodeCur == nodeDest)
  BacktracePath (nodeCur);
  return TRUE;
 end if

 MarkNode (nodeCur);

 for each neighbor nodeNbr of nodeCur
  if (not Marked (nodeNbr))
   W^path2nbr = Aggr (W^path2cur, Best(W^cur2nbr))

   if (W^path2nbr ≺ C^E2E)
    if (MCP(nodeNbr, nodeDest, W^path2nbr, C^E2E))
     BacktracePath (nodeCur);
     return TRUE;
    end if
   end if
  end if
 end for

 UnmarkNode (nodeCur);
 return FALSE;
```

Figure 6. Use of the new operators in a path finding algorithm

the node in the path between two endpoints. Then the value $TRUE$ is returned, which signals that a path with acceptable properties has been found.

If the end node is not yet reached, the $nodeCur$ is marked with the help of function $MarkNode$. This is a common practice in DFS-algorithms, which aims to prevent loops. In the following $foreach$ loop all neighbors of $nodeCur$ are considered that have not been marked. For each neighbor $nodeNbr$ a weight $\overrightarrow{W}^{path2nbr}$ of an path between start and $nodeNbr$ nodes is computed. Corresponding to Section IV-A, the best possible value of the considered segment weight $\overrightarrow{\overrightarrow{W}}^{cur2nbr}$ is aggregated with the intermediate sum $\overrightarrow{W}^{path2cur}$. If the computed weight of the new intermediate path still better than E2E-constraint $\overrightarrow{C}^{E2E}$, the MCP function is called recursively. This time, $nodeNbr$ is used to mark the end of the intermediate path. If the function returns TRUE, the node is saved in order to back trace the path; subsequently TRUE is returned. If the call to the MCP function was not successful, the next neighbor has to be considered likewise. If all neighbors have been considered without any success, the node $nodeCur$ is unmarked and the value $FALSE$ is returned.

Please note that for the sake of simplicity in this algorithm at most one connection between two nodes is supported. An

extension for multigraphs would require an additional loop for all edges between two interconnected nodes. Furthermore, also the back tracking function should be extended in this case, in order to track not only nodes along the path but also along used edges.

VI. Conclusion and Future Work

In this paper, we have defined a novel schema for the generic treatment of network connection properties. In order to support operations on arbitrary properties of network connections, we propose to associate five functions with the ID of every supported property. These functions are summarized in Table I. Three of these functions, which are used for property aggregation and comparison, are mandatory. The mandatory function _AGGREGATE_LINKPART is dedicated to compute the property of connection based on only partial views at the same inter-domain connection. For elaborated discussion about its necessity we refer to [2]. The remaining selection functions aim to simplify handling with value ranges. These functions are not mandatory, as they can be easily derived based on comparison function.

Function class	Purpose
_COMPARE	Compare two values *a* and *b*. Result can be: "*a* is better", "*a* is worse", "*a* and *b* are equivalent"
_SELECT_BEST	Optional function returning the best value of a given value set
_SELECT_WORST	Optional function returning the worst value of a given value set
_AGGREGATE_LINKS	Aggregate property values of two links or paths
_AGGREGATE_LINKPARTS	Aggregate two partial views at the same link to a single link weight

Table I
FUNCTIONS FOR OPERATIONS ON A SINGLE QOS PARAMETER

Together with [2] and [3], which present an information model and a multi-domain routing procedure, the solution presented here is an integral part of our ongoing work, which enables user-tailored connection services with guaranteed E2E quality. However, the generic operation handling proposed in this paper is not restricted to only our work and can be used in alternative routing algorithms that are considering multiple properties, such as [5] and [6].

The presented proposal leaves some aspects unsolved; they will be addressed in our further research as follows: In the first place, a meta-language for the description of property-related functions has to be selected; also, a structure for the registration tree has to be proposed. In order to achieve this, a profound evaluation of alternatives is needed. In the case that a single global registration tree has to be used by multiple organizations, like it is the case for the internet registration tree, the description of equivalence relationships between different entries has to be addressed. Furthermore, the quality parameters of different network layers as well as user-faced services depend on the quality of the underlying layers they are realized upon. Therefore, a general description of such interdependencies and parameter transformations is essential in order to offer customer-demanded quality based on network-specific information.

Acknowledgment

The authors wish to thank the members of the Munich Network Management Team (MNM Team) [10] for fruitful discussions and valuable comments on previous versions of this paper. The MNM Team directed by Prof. Dr. Dieter Kranzlmüller and Prof. Dr. Heinz-Gerd Hegering is a group of researchers at Ludwig-Maximilians-Universität München, Technische Universität München, the University of the Federal Armed Forces and the Leibniz Supercomputing Centre of the Bavarian Academy of Science.

References

[1] CERN, *LHC - The Large Hadron Collider Homepage*, [Online: http://lhc.web.cern.ch/lhc/], August 2010.

[2] M. Yampolskiy, W. Hommel, P. Marcu, and M. K. Hamm, *An information model for the provisioning of network connections enabling customer-specific End-to-End QoS guarantees*, Proceedings 7th IFIP/IEEE International Conference on Services Computing (SCC 2010), pp. 138–145. Miami, 2010.

[3] M. Yampolskiy, W. Hommel, B. Lichtinger, W. Fritz, and M. K. Hamm, *Multi-Domain End-to-End (E2E) Routing with multiple QoS Parameters. Considering Real World User Requirements and Service Provider Constraints*, The Second International Conference on Evolving Internet (INTERNET 2010). Valencia, 2010.

[4] F. A. Kuipers, *Quality of service routing in the internet: Theory, complexity and algorithms*, PhD thesis. Delft University Press, 2004.

[5] T. Korkmaz and M. Krunz, *Multi-constrained optimal path selection*, Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), pp. 834–843. 2001.

[6] P. Van Mieghem, H. De Neve, and F. A. Kuipers, *Hop-by-hop quality of service routing*, Computer Networks, pp. 407–423. Elsevier, 2001.

[7] M. Ziegelmann, *Constrained Shortest Paths and Related Problems*, PhD thesis. VDM, 2007.

[8] R. Bellman, *The theory of dynamic programming*, Proceedings of the National Academy of Sciences of the United States of America, pp. 716–719. 1952.

[9] G. Bertrand, S. Lahoud, M. Molnar, and G. Texier, *Inter-Domain Path Computation with Multiple Constraints*. 2008.

[10] *Munich Network Management Team (MNM Team) Homepage*, [Online: http://www.mnm-team.org], August 2010.