Rerouting Strategies for Networks with Advance Reservations

Lars-Olof Burchard, Barry Linnert, Joerg Schneider
Technische Universitaet Berlin
{baron,linnert,komm}@cs.tu-berlin.de*

Abstract

Network transmissions in high performance networking scenarios, e.g., used for e-science or Grid applications, require quality-of-service guarantees concerning bandwidth availability, but also timing constraints, e.g., deadlines, must be met. Current research efforts concentrate on supporting such environments with SLA-aware advance reservation management systems. Hence, the robustness of the management system against network failures is an important issue, especially since failures frequently occur in networks. Since accurate knowledge about the failure duration is unlikely available and estimations lead to considerably degraded performance, in this paper we present a novel load-based approach for dealing with link failures in advance reservation environments. The approach does not rely on prediction of the downtime, but instead reroutes flows only based on available information about the network.

1 Introduction

In contrast to *immediate reservations*, where the allocations are established immediately after the request is admitted, e.g., using RSVP or other reservation protocols, *advance reservations* allow to specify and request a given quality-of-service (QoS) for a resource a long time before the actual usage starts. Advance reservations are especially useful in environments that require reliable allocations of various resource types at different locations. Such *co-allocations* are necessary in order to assure that all the resources required are available at a given time.

One particular application environment for advance reservations is Grid computing. Many scientific applications, e.g., in the areas of bioinformatics, astronomy, or particle physics, work on large amounts of data using world-wide distributed computing facilities, e.g., supercomputers. In these application environments, the tremendous amounts of computing and network capacity cannot be provided at a single site, in particular when the data source(s), e.g., particle colliders or radio telescopes, are located at very different places. For example, a single radio telescope may generate a data stream of up to 4 GBit/s [5]. Even in todays research networks, the available capacity may not be sufficient to guarantee several such applications may run simultaneously. In order to fulfill the requirements of such e-science applications, computational resources around the globe are interconnected with the data sources using high capacity circuit switched optical networks. This setting provides a closed network environment with a dedicated fibre exclusively available for the collaboration on e-science applications, called *LambdaGrid* [5]. As the resource usage, e.g., of radio telescopes, is usually very expensive, it is necessary to develop mechanisms which guarantee access to the network and other required facilities in a reliable manner, in the Grid scenario this means reserving all the necessary resources in advance. Therefore, the network connections within the LambdaGrid, i.e., switched circuits, must be allocated using advance reservations to assure timely availability of the required bandwidth.

As network failures occur frequently in computer networks [7], recovery strategies for these cases are required especially in SLA-aware advance reservation environments and must consider the special requirements and features of the allocation mechanism. Thus, using knowledge about future network usage is important. Previous work [3] showed that in this context it is useful to reroute also flows that are not yet started but likely to be affected by a failure since they are planned on a failed link and start before the network resource is recovered. Based on this observation, a key issue is to determine when and how flows must be rerouted.

In the ideal case, accurate knowledge about the failure duration is available. However, such information

*This work was partly supported by a grant from Cisco Systems.

is difficult to obtain and therefore predictions must be made. As shown in [3], in such a case underestimations of the actual duration of the failure lead to an aggravation of the impact of the failure, as too few flows are rerouted. Therefore, in this paper strategies are developed and compared which do not rely on predictions of the downtime but rather use the actual load situation on the network in order to determine which flows are considered for rerouting. With this approach, only those flows are taken into account whose rerouting cannot be delayed. Flows starting at later times can be considered to be still safely rerouted in subsequent stages of the recovery process.

This load-based rerouting approach was embedded in an advance reservation management system using the bandwidth broker framework presented in [3] and evaluated using simulations. We present a basic variant which outlines the general approach and an improved feedback-based version which requires no manual adjustments of any parameters, showing that our approach leads to similar results than possible with accurate knowledge about failures while having the significant advantage that knowledge about failure durations is not required.

In the following sections, we describe briefly the application environment for advance reservations and the basic failure recovery mechanism. After that, the rerouting approaches which determine how flows are considered for rerouting are described. In Sec. 5, the downtime-independent strategies are evaluated under different conditions using extensive simulations. The paper concludes with a discussion of related work and some final remarks.

2 Advance Reservation Environment

In this section, the properties of advance reservations and the framework for failure recovery of advance reservations are outlined. As this has already been described in [3], we restrict ourselves here to a brief discussion of the most important aspects.

2.1 System Model

Advance reservations in our case are requests for a certain bandwidth during a specified period of time. In general, an advance reservation is distinguished from an immediate reservation by the time interval the resources are booked in advance. In our notion, this interval is called *reservation time*. The time period for which resources can be reserved in advance is called *book-ahead time* t_{ba}. In this paper, time is divided into *slots* of fixed size [10], each slot covering a predefined interval, e.g., one minute.

We assume the use of the same management system as described in [2, 3], i.e., a *bandwidth broker* as admission control and failure recovery component for a network domain. Flows are admitted by the bandwidth broker and then assigned to a network path. Such advance reservations require a different approach for dealing with link failures. In immediate reservation environments, a link failure affects mainly the flows that are active at the time the failure occurs. Future flows are not affected, because they are not yet admitted by the network management system. In contrast, due to the nature of advance reservations, link failures in such environments do not only affect currently active flows but also those that have already been admitted but are not yet active (see Fig. 1). Since a QoS guarantee, i.e., the start and stop time together with the requested bandwidth, has already been given for admitted flows, interruptions of these flows must also be avoided. We refer to this process as *rerouting in advance*. Rerouting those inactive but affected flows too late causes considerably a higher termination rate [3].

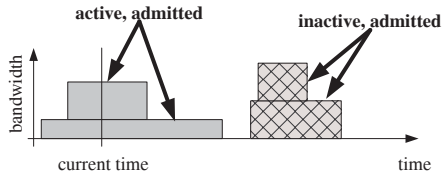

Figure 1. Flows in Advance Reservation Environments

2.2 Failure Recovery Process

In this section, the general failure recovery mechanism is briefly described, for more details, please refer to [3].

Once a failure on the network is notified, e.g., reported to the bandwidth broker, the following steps are employed: first, the set of affected flows, i.e., those to be rerouted, are determined. This is achieved using the downtime-independent strategy presented in this paper. In a second step, the affected flows are mapped onto alternative paths. This second step is discussed on detail in [3].

Using the system and failure recovery mechanism as given above, two straightforward strategies are conceivable in order to tackle the recovery. Consider the recovery process at a given time t. At the beginning of this slot, it is possible that only the flows that will

commence during $[t, t+1]$ are rerouted. This strategy, in the following referred to as A_1 does not unnecessarily reroute flows, i.e., only flows affected by the failure are handled. Thus, A_1 behaves like an immediate reservation approach [1]. In the following, we refer to the amount of unnecessarily rerouted flows as *overhead*. The second strategy is to reroute any admitted flow on the failed link, independent of its start time. This strategy (A_{all}), needs to be carried out only once at the beginning of the failure and very likely reroutes flows although they will never be affected by the failure, in particular in cases, where the failure lasts only a short period of time. At the other hand, A_{all} achieves the lowest possible amount of terminated, admitted, but not yet started, flows, as they are rerouted at the earliest possible time. This cannot be assumed for A_1, as flows are rerouted immediately before they are to start. At this time, the remaining alternative routes are likely congested and do not provide sufficient bandwidth.

In order to achieve the optimal results with respect to both the number of termination flows and unnecessarily rerouted flows, an oracle is needed which provides the exact length of the downtime in the moment the failure occurs. Using this information, only the flows within the downtime will be considered for rerouting ($A_{optimal}$). This produces neither a high number of terminated flows nor overhead. However, $A_{optimal}$ is only theoretically applicable as the downtime of a link is usually unknown. Substituting the exact knowledge of the downtime by vague estimations is dangerous, since this leads to a significantly worse performance [3].

3 Downtime-Independent Rerouting

In this section, we present our novel downtime-independent rerouting approaches. Two variants are given which rely on knowledge about the load on the network (*load-based*) or use information obtained during the rerouting process (*feedback-based*), respectively. In addition to presenting the algorithms, we show how they can be applied in centralized, e.g., bandwidth brokers, and distributed environments.

3.1 Outline

The general approach followed by our downtime-independent recovery strategies is to identify and reroute flows which are unlikely to be safely rerouted at any later point in time. For that purpose, in each time slot throughout the duration of the link failure a *rerouting interval* is calculated. The length of this rerouting interval, is *independent* of the actual downtime which is unknown in the system.

All flows which use the broken link for at least one time slot of the rerouting interval will be tried to reroute for these affected time slots. If no alternative path [3] provides sufficient bandwidth, the flow will remain on the broken link until its start time since the failure may have ended until then, otherwise the flow will be terminated then. All other flows will not be rerouted, even if their currently assigned route contains the broken link. With a look at the example in Fig. 2,

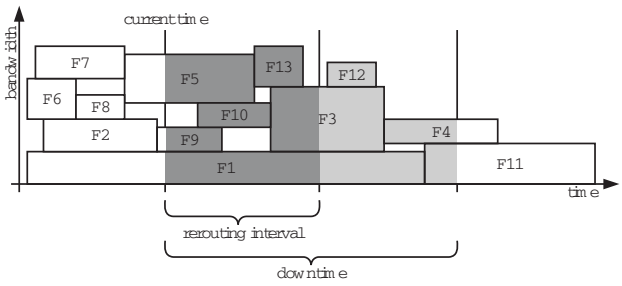

Figure 2. Example: Downtime, rerouting interval, and affected flows.

this means that the dark gray flows must be rerouted and the light gray flows will be left on the current link for the moment. At a later time slot, F12 and additional parts of F1 and F3 may also be within the new rerouting interval and thus, must be rerouted.

The broken link will be blocked for the rerouting interval only. This means, new reservations for time slots after the rerouting interval may be booked on the currently broken link and will be rerouted later if necessary.

3.2 Objectives for the Rerouting Interval

It was shown in [13] that in advance reservation environments a "critical time" can be determined from where on the probability of successful admission increases significantly. This can be mapped onto failure recovery scenario, i.e., rerouting flows reserved for the far future reduces the number of terminated flows, i.e., the rerouting interval must be long.

On the other hand, if the rerouting interval is longer than one time slot, it is possible that more flows than necessary are rerouted to other resources, i.e., the overhead increases. Reducing the overhead is crucial in SLA-aware environments, i.e., if fees must be paid for the allocation of alternative links. Furthermore, initially flows are mapped on the shortest available path in order to avoid blocking too many links [11]. Thus, routing flows on alternative, i.e., generally suboptimal paths, blocks too many links and leads to a reduced

network utilization. Hence, it is necessary to determine the rerouting interval as short as possible to keep the overhead low.

As both requirements are contrary, a trade-off must be made between both metrics. In SLA-aware environments, this trade-off is designated by the fine to pay for terminated flows and the prices to pay for the alternative links used for the rerouted flows.

3.3 Load-Based Rerouting

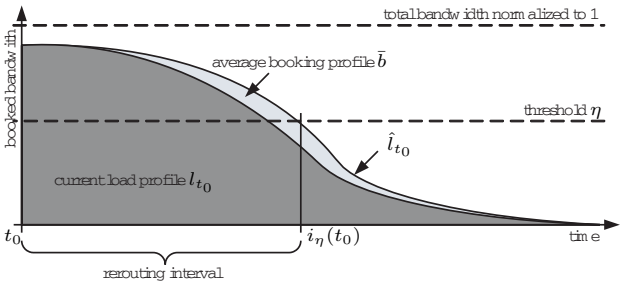

Figure 3. Determination of the rerouting interval based on the combination of the current load profile and the average booking profile.

The load-based approach bases on the *critical time* [13] in order to determine the remapping interval length.

The load situation of a time slot t_0 is described by a *load profile* $l_{t_0}(t)$, with $t \in \mathbb{N}$, which is defined as the normalized totally allocated bandwidth on all links for each future time slot $t_0 + t$. The booking behavior is described by the average booking profile. The *booking profile* $b_{t_0}(t)$, with $t \in \mathbb{N}$, of a time slot t_0 is the normalized booked bandwidth per future time slot $t_0 + t$ of all incoming reservations during this time slot. The *average booking profile* $\bar{b}(t)$ is the average over all previous booking profiles. The load and booking profiles are calculated considering the whole network with all links. Selecting only the links of the network which may have an impact on the rerouting is difficult and time-consuming, as all alternative paths of the flows currently booked on the broken link must be examined.

The following algorithm is used to calculate the rerouting interval based on the load. Firstly, a *combined profile* $\hat{l}_{t_0}(t), t \in [t_0, t_{ba}]$ is created for the current time slot t_0 based on the current load profile $l_{t_0}(t)$, combined with the average booking profile $\bar{b}(t)$. Adding both profiles gives an indication of the expected network load after this time slot as it sums up the currently booked load and the average incoming load during one time slot. The rerouting interval $[t_0, t_0 + i_\eta(t_0)]$

```
1 link failed at t_0
2 while link down do
3  // calculate the profiles
4  init l̂_{t_0}
5  for each flow f ∈ F, f.start > t_0 do
6     for each t ∈ [f.start, f.stop] do
7        l̂_{t_0}(t) += f.bandwidth
8  add average booking profile b̄ to profile l̂_{t_0}
9  // calculate i
10 for i = t_ba to 1 do
11    if not l̂_{t_0}(i) < η then break
12 // reroute flows within rerouting interval i
13 for each f ∈ F* do
14    if f.start < t_0 + i ∧ f.stop > t_0 then
15       reroute f for [f.start, t_0 + i]
16 wait for next time slot: t_0 = t_0 + 1
```

Figure 4. Schematic overview of the algorithm for the calculation of the rerouting interval and the load-based rerouting.

is determined using a *threshold* η with $\eta \in [0, 1]$. The threshold η can be predefined and is related to the critical time [13]. The rerouting interval is defined by the time after which all values of the combined profile are lower than η (see Fig. 3):

$$i_\eta(t_0) := \min\left\{ i \,|\, \forall t > i \,:\, \eta > \hat{l}_{t_0}(t) \right\}$$

The whole set of flows on the network is denoted by F, and the set of flows mapped onto the failed link is denoted by F^*. Each flow f using a failed link has assigned three properties f.bandwidth, f.start, and f.stop.

As shown in the schematic overview in Fig. 4, the rerouting interval, defined by $i_\eta(t_0)$, is computed once for each time slot as long as the failure lasts. Alike the critical time defined in [13], the exact definition of the threshold η is subject to the actual network setting, e.g., topology.

3.4 Feedback-Based Rerouting

The feedback-based approach modifies the previously shown load-based procedure in using the feedback, i.e., computing the probability for successful rerouting, obtained during the rerouting process.

The respective section of the rerouting algorithm is modified as depicted in Fig. 5. In the initial step, i.e., the time slot when the failure is detected, the rerouting interval is increased step by step until the success

```
// calculate i
if current time==t_0 then
    i = t_0 + 1 // initial interval computation
else
    i = î // save value from last iteration
do
    reroute flows within [t_0 + i, t_0 + i + 1]
    p := rerouting probability for [t_0 + i, t_0 + i + 1]
    i := i + 1
while p < 0.9
î := i // save for next time slot
wait for next time slot: t_0 = t_0 + 1
```

Figure 5. Schematic overview of the feedback-based algorithm.

probability is sufficiently high. When advancing to the next time slot, the same procedure applies starting from the rerouting interval determined in the previous step ($\hat{i}$). Thus, the length of the rerouting interval during a single failure never decreases, in particular the initially computed interval marks the minimal value. The advantage of the feedback-based approach is the self-adjusting rerouting interval even in case of highly varying load situations without needing to adjust η. The feedback method can also be combined with external knowledge about the failure duration thus restricting the rerouting interval.

4 Application in Distributed Environments

The application of the previously outlined strategies in a centralized environment, e.g., a bandwidth broker controlled network domain, is straightforward. In this case, the load or feedback information available in the broker is used as input for the failure recovery module of the broker. The bandwidth broker is considered as the instance which performs the rerouting as described in [2, 3].

In general, in distributed scenarios either *path protection* or *link protection* strategies are conceivable, e.g., assuming an MPLS-based environment with route-pinning opportunity. In this case, upstream edge routers (path protection) or routers adjacent to the failed link (link protection) are notified of link failures and upon this notification individually initiate the setup of new routes for the affected flows locally known [1]. In an advance reservation scenario, such approach can also be implemented, e.g., using the distributed architecture outlined in [12].

In the distributed scenario, our algorithms can also be applied in a straightforward manner on routers responsible for rerouting, e.g., edge routers. These routers are then only responsible for rerouting the flows locally known, propagating path updates to other routers [12]. The feedback-based approach can be implemented on the edge routers without further modification, i.e., edge routers use the feedback obtained during rerouting the flows entering the network on the router to obtain the length of the rerouting interval. The load-based approach is also applicable, although the obtained load information may be inaccurate, e.g., when only the load situation on the edge router itself is used.

In a link protection scenario, the identical procedure applies with the difference that instead of the edge routers, the routers immediately preceding the failed link carry out the failure recovery mechanism.

5 Evaluation

In this section, the experimental setup and the results of the evaluations are described.

5.1 Simulation Environment

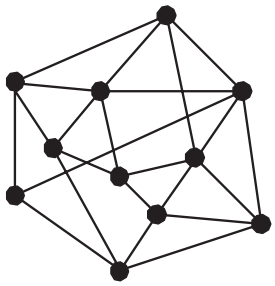

Figure 6. Network topology

The different strategies were evaluated using a simulation environment for advance reservations. The network topology as depicted in Figure 6 was used, representing an actual ISP backbone network with 1000 MB/s capacity for each link. Different load situations on the network were simulated using different values for the average reservation time r (see Sec. 2.1).

We used a failure model with a failure duration of 500 slots in order to point out the effect of the rerouting strategies, which allows to study the impact of applying our the strategies on the network performance without dealing with side-effects of a more complex model.

The source and destination nodes of flows were randomly chosen, with the same probability for each node

of the network, the duration of flows was uniformly chosen within $[0, 200]$ slots, and the bandwidth requirement was uniformly distributed among 56, 64, 128, 256, 512, and 1024 kB/s. For most simulations, the network was equally loaded, i.e., server and client nodes were uniformly distributed among the network. However, also an unequally loaded network was simulated which means that in this case only 6 nodes represent servers.

Each simulation run had a duration of $20,000$ slots. In order to obtain meaningful results, the simulations were repeated until a sufficiently small confidence interval (within $\pm 1\%$ of the mean with 95% confidence) was reached. Finally, the two metrics used to assess the performance of the load-based approach are given as follows:

$$\text{termination ratio} := \frac{|T|}{|A|}, \qquad \text{overhead} := \frac{|O|}{|O| + |A|},$$

with T and A being the set of terminated and the set of affected flows, respectively, and O being the overhead (see Sec. 2.2), i.e., the set of flows that are remapped although not actually affected by the failure ($T \subseteq A \subseteq F^*$, $O \subseteq F^*$, see Sec. 3.3). Due to the space limitation, we restrict ourselves to a simulation of the centralized version of the algorithms, i.e., using the bandwidth broker.

5.2 Load-Based Rerouting

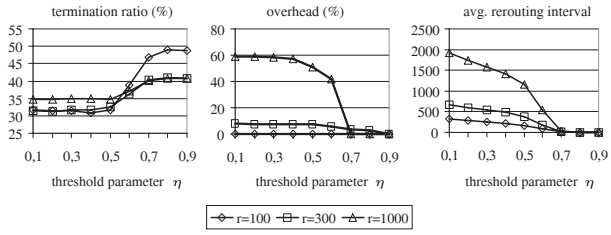

Figure 7. Termination ratio (*left*), overhead (*center*), and length of the rerouting interval (*right*) for varying parameter r.

In order to outline the general relation between η, remapping interval length, and the performance in Fig. 7 the termination ratio, the overhead, and the average length of the rerouting interval are depicted depending on the choice of the average reservation time r. With $\eta \leq 0.5$, the termination ratio remains at a constantly low level and rises significantly for $\eta > 0.5$. The opposite effect can be seen for the overhead. As depicted, this holds for any of the chosen values of r. The trade-off between termination ratio and overhead, i.e., the actual choice of η can be made by the network operator, depending the actual network parameters, e.g., network load and topology. As shown in Fig. 8, the feedback-based approach always reaches the performance of the best choice of η. Fig. 8 shows

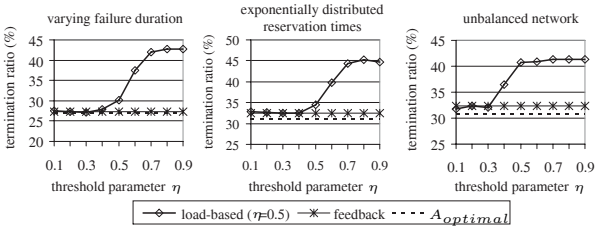

Figure 8. Termination ratio for varying failure duration (*left*), exponentially distributed reservation times (*center*), and unevenly loaded network (*right*), with mean $r = 300$. The dotted line denotes the results when having exact knowledge about the failure duration ($A_{optimal}$).

the termination ratio of both the feedback-based and the load-based approach with varying parameter η. It can be observed that the feedback-based approach is stable under diverse conditions and always reaches the performance using the optimal value of η. With varying failure duration (normally distributed with mean 500), with exponentially distributed reservation times (with mean $r = 300$), and also when the network is not evenly loaded – in our case this means that only 6 nodes represent servers – the actual performance of the load-based approach does not differ to the previously shown results. This underlines the success of our strategy under various different conditions.

5.3 Comparison of the Algorithms

In Fig. 9, the performance of the load-based approach with $\eta = 0.5$, i.e., where the minimal termination ratio is reached, and of the feedback-based approach is compared to the result when using the algorithms described in Sec. 2.2. Again, constant failure duration of 500 slots was used with $r = 300$. In terms of the termination ratio – assumed to be the more important performance metric – the feedback-based approach achieves the same results as the load-based approach and both results are comparable to the "oracle" algorithm with exact knowledge, i.e., $A_{optimal}$.

The most important result is the termination ratio for the scenario with $r = 100$. In this case, although

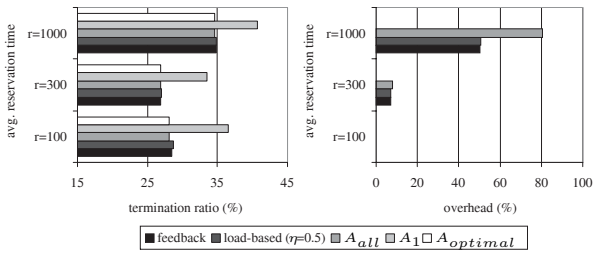

Figure 9. Termination ratio and overhead of the load-based ($\eta = 0.5$) and the feedback-based approach compared with the algorithms A_1, A_{all}, and $A_{optimal}$.

the rerouting interval is much shorter than the failure duration, the downtime-independent approaches still reach the optimal result, i.e., the same performance as possible with exact knowledge. This is achieved with overhead of nearly zero.

6 Related Work

The general properties of advance reservation mechanisms in computer networks have already been widely discussed [6, 14, 10]. The authors of these papers examined fundamental properties and the difference compared between advance and immediate reservations. In [12], a distributed framework for dealing with advance reservations was dicussed with the focus on distributing and replicating knowledge about advance reservation requests in order to be less sensitive to failures of admission control instances.

Using the actual network load for selecting the period of time where flows are rerouted has its origin in the observation that a certain *critical time* [13] exists in advance reservation systems, which defines the point in time from whereon reservations are accepted with very high probability. This is independent of the distribution of requests, reservation times, etc. In our framework, the rerouting interval, defined by the threshold parameter η, is related to this critical time.

The considerations described here are based on earlier work [3], showing the advantages of rerouting also flows that are not yet started but likely to be affected by a network failure. The limitations of the approach to estimate the actual duration of a failure were also shown, in particular underestimations lead to a significantly higher termination ratio compared to an ideal situation where the duration of the failure can be accurately predicted. The approach presented in this paper overcomes this limitation by repeatedly computing the rerouting interval based on the actual network load.

In [9], a load balancing and fault tolerance mechanism for immediate reservations based on MPLS aware network infrastructure was discussed. The main idea was to distribute the flows or packets of a flow onto several *maximally disjoint paths* in order to more evenly balance the network load and reduce the impact of a link failure onto the flows. Therefore, a link failure affects only a part of the flow which is then rerouted using one of the other available paths.

In order to further increase the success of the failure recovery strategy, a number of other approaches are conceivable which are not related to the computation of the affected flows but deal with how to reroute. In this context, the approach described in [4] deals with preempting lower priority flows in order to avoid more costly terminations of flows with high priority. The authors present a heuristic for the computation of the flows to be preempted such that rerouting of flows is minimized. Approaches like this can easily be integrated into our framework.

7 Conclusion

In this paper, we presented a novel failure recovery approach for advance reservation scenarios, using only the available information at the time of the failure for the selection of a *rerouting interval* instead of requiring estimations about the failure duration. Two variants were presented, using the actual network load and feedback obtained during the rerouting process, respectively. A particular application is the bandwidth broker described in [2] on the network level and the VRM [8] on the Grid level, where the co-allocation of these different resource types demands also for appropriate failure recovery strategies. However, our strategies are also applicable in distributed settings although these are less realistic target platforms for advance reservations. The simulations showed the feedback-based variant achieves similar results while avoiding the need to adjust the threshold parameter η.

Future work will, e.g., concentrate more on the application of the algorithms for environments with distributed bandwidth broker.

References

[1] A. Authenrieth and A. Kirstaedter. Engineering End-to-End IP Resilience Using Resilience-Differentiated QoS. *IEEE Communications Magazine*, 1(1):50–57, January 2002.

[2] L.-O. Burchard. Networks with Advance Reservations: Applications, Architecture, and Performance. *Journal*

of Network and Systems Management, 2005. to appear.

[3] L.-O. Burchard and M. Droste-Franke. Fault Tolerance in Networks with an Advance Reservation Service. In *11th International Workshop on Quality of Service (IWQoS), Monterey, USA*, volume 2707 of *Lecture Notes in Computer Science (LNCS)*, pages 215–228. Springer, 2003.

[4] J. de Oliveira, C. Scoglio, I. Akyildiz, and G. Uhl. A New Preemption Policy for DiffServ-Aware Traffic Engineering to Minimize Rerouting. In *IEEE INFOCOM, New York, USA*, pages 695–704, 2002.

[5] T. DeFanti, C. de Laat, J. Mambretti, K. Neggers, and B. S. Arnaud. TransLight: A Global-Scale LambdaGrid for E-Science. *Communications of the ACM*, 46(11):34–41, November 2003.

[6] D. Ferrari, A. Gupta, and G. Ventre. Distributed Advance Reservation of Real-Time Connections. In *Network and Operating System Support for Digital Audio and Video*, pages 16–27, 1995.

[7] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of Link Failures in an IP Backbone. In *2nd ACM SIGCOMM Internet Measurement Workshop 2002, Marseille, France*, pages 237–242, Nov 2002.

[8] L.-O. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert. The Virtual Resource Manager: An Architecture for SLA-aware Resource Management. In *4th Intl. IEEE/ACM Intl. Symposium on Cluster Computing and the Grid (CCGrid), Chicago, USA*, 2004.

[9] S. Lee and M. Gerla. Fault-Tolerance and Load Balancing in QoS Provisioning with Multiple MPLS Paths. In *Proceedings of IFIP Ninth International Workshop on Quality of Service (IWQoS)*, 2001.

[10] M. Degermark, T. Köhler, S. Pink, and O. Schelen. Advance Reservations for Predictive Service. In *5th Intl. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV), Durham, USA*, volume 1018 of *Lecture Notes in Computer Science (LNCS)*, pages 3–15, 1995.

[11] Q. Ma and P. Steenkiste. On Path Selection for Traffic with Bandwidth Guarantees. In *5th IEEE International Conference on Network Protocols (ICNP), Atlanta, USA*, pages 191–204, 1997.

[12] S. Bhatnagar and B. Nath. An Edge Router Based Protocol for Fault Tolerant Handling of Advance Reservations. In *IEEE Intl. Conference on Communications (ICC), New York, USA*, pages 1086 –1093, 2002.

[13] D. Wischik and A. Greenberg. Admission Control for Booking Ahead Shared Resources. In *IEEE INFOCOM, San Francisco, USA*, pages 873–882, 1998.

[14] L. Wolf and R. Steinmetz. Concepts for Resource Reservation in Advance. *Kluwer Journal on Multimedia Tools and Applications*, 4(3):255–278, 1997.