Performance Issues of Bandwidth Reservations for Grid Computing

Lars-Olof Burchard, Hans-Ulrich Heiss
Communication and Operating Systems Group
TU Berlin, Germany
{baron,heiss}@cs.tu-berlin.de

César A. F. De Rose
Catholic University of Rio Grande do Sul (PUCRS)
Post-Graduate Program in Computer Science, Porto Alegre, Brazil
derose@inf.pucrs.br

Abstract

In general, two types of resource reservations in computer networks can be distinguished: immediate reservations which are made in a just-in-time manner and advance reservations which allow to reserve resources a long time before they are actually used. Advance reservations are especially useful for grid computing but also for a variety of other applications that require network quality-of-service, such as content distribution networks or even mobile clients, which need advance reservation to support handovers for streaming video. With the emerged MPLS standard, explicit routing can be implemented also in IP networks, thus overcoming the unpredictable routing behavior which so far prevented the implementation of advance reservation services. The impact of such advance reservation mechanisms on the performance of the network with respect to the amount of admitted requests and the allocated bandwidth has so far not been examined in detail. In this paper, we show that advance reservations can lead to a reduced performance of the network with respect to both metrics. The analysis of the reasons shows a fragmentation of the network resources. In advance reservation environments, additional new services can be defined such as malleable *reservations which are introduced in this paper and can lead to an increased performance of the network. Four strategies for scheduling malleable reservations are presented and compared. The results of the comparisons show that some strategies increase the resource fragmentation and are therefore unsuitable in the considered environment while others lead to a significantly better performance of the network. Besides discussing the performance issue, in this paper the software architecture of a management system for advance reservations is presented.*

1 Introduction

Grid computing applications require quality-of-service (QoS) guarantees on various fields. For example, the allocation of computing resources (processor nodes) is made using *advance reservation* mechanisms, i.e., the processors are allocated a long time before they are actually used. In order to also allocate network resources in the same way, it is required to implement the corresponding advance reservation mechanisms on computer networks.

In the field of advance reservations in computer networks, although some work has been carried out - especially architectural considerations and implementational aspects [6, 7, 16] - many important aspects have not been studied, yet. Especially the relation of advance reservations to the performance of the network, e.g., measured by the admission probability or the throughput, i.e., the amount of bytes carried by admitted flows, has so far not received much attention. However, in order to implement an advance reservation service which needs a considerable amount of stability and predictability in terms of routes and status, a network infrastructure is required which overcomes the drawbacks currently associated with IP networks such as unpredictable routing behavior. Therefore, the *multi protocol label switching* (MPLS) [11] architecture can be seen as an interesting new approach for supporting advance reservations in computer networks. MPLS provides mechanisms for traffic engineering, in particular *explicit routing*, and rerouting in case of link failures. These two functionalities are ideally suited to support advance reservations. The MPLS technology allows to consider advance reservations as a feasible way of allocating network bandwidth. Important issues in such an environment are the admission control procedure and how reservations are set up on the network. For that purpose, we consider a bandwidth broker that acts as network management system, i.e. grants and denies access to the network, and initiates the explicit routing process.

So far, a comparison of the properties of both reservation types regarding their impact on the network performance has not been made. The first part of this paper deals with such a performance comparison and shows that the definition of advance reservations can result in a decreased call acceptance rate of the network while the bandwidth blocking rate increases. In an advance reservation environment, the available information about future network utilization allows to define additional new services. In the second part of the paper we show how the definition of *malleable reservations*, i.e. reservations without fixed timing and bandwidth requirements, allow to improve the performance in advance reservation environments. For those reservations, the network management system tries to find a suitable time for the requested transmission and hence releases the client from this task. Defining malleable reservations can be of interest for network users since it increases the probability that requests are admitted. Network operators also benefit from allowing the definition of malleable reservations, since this reduces the bandwidth blocking rate of the network. In this document, a basic admission control algorithm which requires polynomial time is presented together with four variants which use different strategies to schedule malleable requests. Such functionality cannot be implemented in a similar way in an immediate reservation environment. These mechanisms were integrated into a management system for advance reservation described in Section 6.

2 Related Work

Early works considering advance reservations concentrated on the basic requirements to facilitate such mechanisms, e.g. call admission control [6], while others developed architectures such as an agent based approach for enabling a scalable advance reservation mechanism [12] based on OSPF routing infrastructure. In contrast to that paper, we assume to use an MPLS [11] aware core network which is controlled by a bandwidth broker.

Many other earlier publications concentrate on extending existing signaling protocols such as RSVP [13] or ST-II [10], or discuss the general framework required to implement an advance reservation service [16].

In [15] an admission control scheme for advance reservations was proposed. The authors also briefly mention the problem that advance reservation mechanisms can lead to a decreased performance in terms of admitted requests. This aspect is examined and discussed more detailed in our paper.

The impact of advance reservation mechanisms on routing problems in a computer network is discussed in [8]. The authors present several algorithms for different problems related to advance reservations such as finding the earliest time for a transmission with given duration and bandwidth and focus on the computational complexity of the algorithms. In contrast to our paper, malleable reservations and performance issues of advance reservations were not discussed.

Advance reservations are particularly important in grid computing environments where not only the allocation of computing resources but also of the corresponding network bandwidth for transmission between different computers involved must be possible in advance. An example for a grid toolkit that support such mechanisms is Globus with its GARA resource allocation component [7]. Another examples for the application of advance reservations is a distributed media server systems as described in [5], where large amount of media files are transmitted between the different servers.

In order to store link status information, the implementations presented in this paper use arrays which were examined and compared with a tree based approach in [4]. The result was, that arrays are faster in the environments considered and are also more memory efficient.

3 Advance Reservations

Before discussing the performance issues, a brief description of the advance reservation environment will be given.

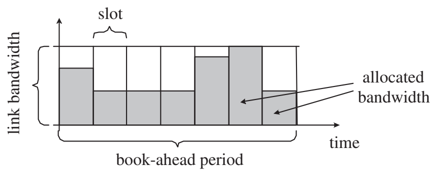

Figure 1. Status information and book-ahead period kept in the network management system for each link

In general, advance reservations differ from immediate reservations only by the time a request is submitted to the network management thus decoupling the submission of the request completely from the usage of the resources. The basic framework has been described in [16] together with a number of issues that must be considered when implementing advance reservations.

In order to perform reliable admission control, status information about the currently known future utilization of links is required. The period for which requests can be submitted is called *book-ahead period* (see Figure 1). Usually, this period is divided into slots of fixed size [4, 12].

The basic parameters to be submitted with a request have been widely discussed, for example in [12, 16]. It is understood, that in an advance reservation environment not only the start of the transmission must be defined but also the duration of the request respectively the stop time of the transmission as outlined in Figure 2. This is required in order to obtain reliable information about the future network status and thus to perform reliable admission control. Consequently, in addition to the source and destination node, the parameters to be defined by a request are:

1. the start time t_{start}
2. the time when the transmission is to be finished t_{stop}
3. the requested bandwidth b (other QoS requirements may also be specified, however in this case we restrict ourselves to bandwidth)

As depicted in Figure 2, this results in two phases: the intermediate phase between the request and the start of the transmission and the actual usage phase where the transmission takes place.

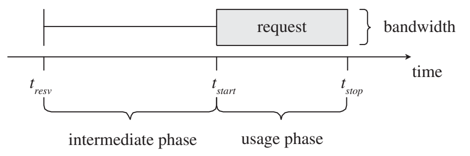

Figure 2. Temporal sequence for advance reservations: requests are submitted at t_{resv}, start at t_{start}, and are finished at t_{stop}

Access to the network is controlled by a management system usually called *bandwidth broker*. A reservation request is submitted to the bandwidth broker at time t_{resv} which is assumed to be considerably earlier than t_{start}, i.e., hours, days, or even more. The bandwidth broker admits only those requests for which sufficient bandwidth can be guaranteed during the requested transmission period.

Other QoS metrics such as delay and jitter are omitted here because bandwidth plays the most important role for applications focused in this paper, which deal mainly with file transfer like transmissions. However, delay or jitter can also be reserved in advance although in terms of complexity the admission control problem becomes NP-complete when more than a single QoS metric must be met. This holds even in the case of immediate reservations [14].

4 Performance Issues

4.1 Simulation Environment

Before showing the results of the performance comparison, in this section the environment in which the tests were made is briefly described.

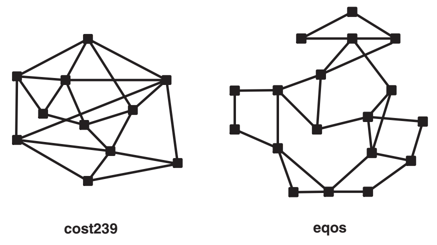

Figure 3. Network topologies

In order to examine the performance of advance reservation mechanisms, we used two backbone network topologies depicted in Figure 3.

The examinations were made using a simulation environment for advance reservations. In order to compare advance and immediate reservations, a set R of requests was generated for each network topology. Each request is defined by its start time, stop time and bandwidth requirement. For the case of immediate reservations, requests are issued at the start time. In order to examine advance reservations, a varying number of requests was randomly chosen from the set and issued a certain amount of time (*reservation time*) before the corresponding transmission is to start.

The book-ahead period had a length of $2^{15} = 32768$ slots. The request lengths were exponentially distributed with a mean value of 100 slots and the bandwidth requirement is uniformly distributed between 64 KBit/s and 1 MBit/s. Each link of the networks was assumed to have a bandwidth capacity of 100 MBit/s. In order to obtain meaningful results which show differences between immediate and advance reservations, a situation with high network load must be simulated, i.e. the call acceptance rate must be below 100%. Otherwise, both reservation types behave identically. The load was unevenly distributed with one third of the nodes being servers and the rest of the nodes clients. For each request, a server and a client was randomly chosen following a uniform distribution.

The performance metrics used for the examinations are call acceptance rate and bandwidth blocking rate in order to assess the effect of the two reservation types on both the amount of accepted calls and the amount of bandwidth that is carried by the network. The bandwidth blocking rate is defined as $\frac{\sum_{r\in A} bandwidth(r)}{\sum_{r\in R} bandwidth(r)}$, where A denotes the set of accepted requests and R denotes the whole set of requests as mentioned before.

4.2 Performance Evaluation

In Figure 4, the influence of a varying percentage of advance reservations on the call acceptance rate and the bandwidth blocking rate of a network is outlined. It can be observed, that even when only a few percent of the reservations are made in advance the performance decreases. The reservations made in advance were randomly chosen from the set R following a uniform distribution.

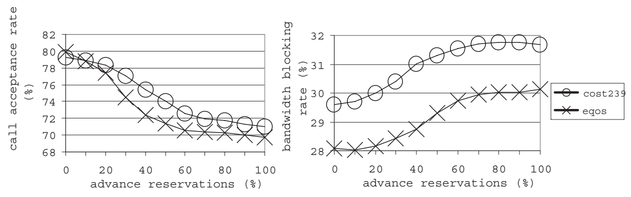

Figure 4. Call acceptance rate and bandwidth blocking rate for different percentage of advance reservations

The performance degradation impacts especially the call acceptance rate but also the bandwidth blocking rate can be affected. This shows, that advance reservations also improve the admission probability for requests with high bandwidth requirement and/or long duration.

The prospective link utilization, i.e. the utilization during the book-ahead period of a certain link as known to the bandwidth broker, at four different points in time $t_1 < t_2 < t_3 < t_4$ shows how advance reservations differ from immediate reservations. In Figure 5, this is illustrated for an arbitrarily chosen link. As more requests are admitted, at certain times the link utilization shows ”peaks” increasing over time.

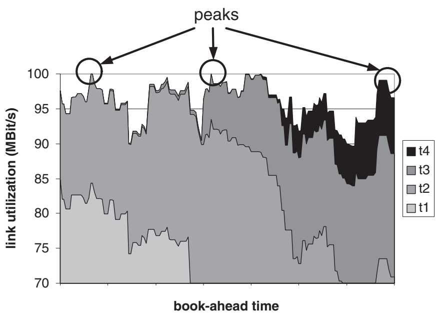

Figure 5. Book-ahead status of a link at four different points in time

4.3 Discussion

In advance reservation scenarios, requests with a large book-ahead can be issued, i.e. requests can be made a long time before the transmission actually takes place. This results in peaks which occur at different times within the book-ahead period and block the bandwidth at those times. This was presented in the previous section. These peaks lead to a fragmentation of the available network resources such that gaps appear which cannot be filled with requests since these gaps are too short for additional requests to fit in.

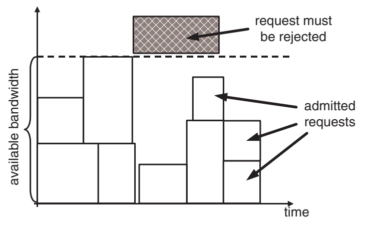

Figure 6. Future requests block available bandwidth

In Figure 6, this effect is outlined. A new request (the dark gray box) cannot be admitted because the gap between two peaks is too short. The gaps that exist between both peaks can become too short for additional requests to fit in. Obviously, such a situation cannot occur in an immediate reservation scenario due to the nature of such a reservation scenarios which does not consider future requests.

5 Malleable Reservations

In this section, we apply the concept of malleable reservations to bandwidth reservation in order to overcome the performance degradation of advance reservations. The idea is to implement a new service for clients that allows to define requests which do not have fixed constraints in terms of start time, stop time, and bandwidth requirement. These reservations are an opportunity to fill the gaps and thus to improve the network performance.

5.1 Properties

In an advance reservation environment, detailed information about the status of each link during the book-ahead period is available. Reservations made in advance include not only knowledge about the start time and the bandwidth requirement but also the stop time of the requested transmission.

Using this information, an approach is to allow reservations being defined without fixed boundaries with respect to the time of start and stop and the bandwidth. In a situation as depicted in Figure 6, e.g. the new request can be admitted when the start and stop times are changed and the bandwidth is increased.

This is useful for a certain type of reservations, e.g. only the total amount of data to be transmitted might be of interest and perhaps a deadline until the transmission must be finished. Examples for such reservations are transmissions of large amounts of data such as backups of data sent to a storage server where the data is written to tapes and the backup must be finished at a certain time. The automatic distribution of large amounts of data in a distributed media server system [5] is another example where such functionality is useful.

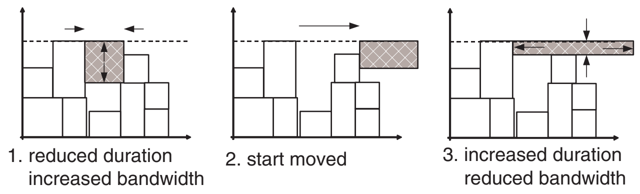
1. reduced duration increased bandwidth
2. start moved
3. increased duration reduced bandwidth

Figure 7. Examples for malleable reservations

We call such requests without fixed parameters but with fixed amount of bytes to be transmitted *malleable reservations*. The actual start and stop time and the bandwidth are not fixed but can be chosen by the network management within a certain range. This range must be defined by the client who issues a request. In Figure 7, three examples are given which show how the rejected request in the situation depicted in Figure 6 can be admitted when being defined as malleable.

5.2 Admission Control

The admission control procedure for malleable reservations and its computational complexity is described in this section.

Usually, in an advance reservation environment a request r can be defined as $r = (u, v, t_{start}, t_{stop}, bw)$, where u, v denote the start and destination node, t_{start}, t_{stop} denote the start and stop time of the transmission and bw the requested bandwidth. In order to distinguish such reservations from malleable reservations, we call them *fixed* reservations. Admission control in this case is to find a feasible path from u to v such that the bandwidth bw is available during the requested transmission period $[t_{start}, t_{stop}]$. An admission control algorithm for the case of using fixed parameters t_{start}, t_{stop}, bw was described in [2].

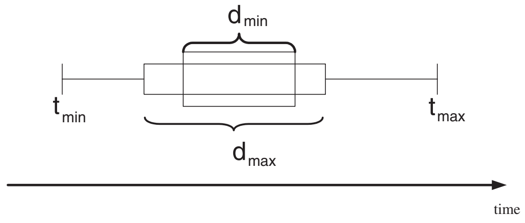

Figure 8. Boundaries for malleable reservations

The admission control algorithm for malleable reservations has to be more complex than required for those with fixed parameters. Requests defining a malleable reservation require to include additional information which describe the boundaries for the request, i.e. an earliest start time or the maximally possible transmission bandwidth.

A malleable request is defined as $r_m = (u, v, t_{min}, t_{max}, d_{min}, d_{max}, c)$, where t_{min}, t_{max} denote the earliest start respectively the latest stop time, d_{min} and d_{max} denote the minimal and maximal duration, and c denotes the total amount of bytes to be transmitted (see Figure 8). Depending on the actual application, instead of defining c it is possible to define other properties for the respective transmission from which the required parameters are then computed by the network management system. For example, for a video streaming application, a maximal bandwidth bw_{max} for the video stream and a fixed duration d might be given. The required parameters then can be computed as $d = d_{min} = d_{max}$ and $c = bw_{max} * d_{max}$. In this document, we assume to use a fixed bandwidth throughout the whole transmission period, i.e. the bandwidth for each admitted request cannot be changed during the transmission. Allowing the bandwidth to vary during the transmission can further increase the network performance, however leads to NP-complete problems as described in [8].

Admission control for malleable reservations requires to find a path with sufficient bandwidth such that the transmission can be made with parameters within the requested boundaries. The following simple algorithm performs admission control for a malleable request $r_m = (u, v, t_{min}, t_{max}, d_{min}, d_{max}, c)$:

```
AdmControl($G(V,E), r_m$)
1 foreach $t_d \in [d_{min}, d_{max}]$ do
2   $bw = \frac{c}{t_d}$;
3   foreach $t_{start} \in [t_{min}, t_{max} - t_d]$ do
4     if (find_path( $u, v, t_{start}, t_{start} + t_d$,$bw$)
5       ==success) then
6         break; //success:  path found
7   done
8 done
```

The algorithm takes as input the network graph $G(E, V)$ and the request r_m, and tests for any possible combination of parameters whether a feasible path, i.e. a path with sufficient bandwidth, exists. If such a path is found, the algorithm stops (line 5). The purpose of the function `find_path` is to determine a path from u to v with sufficient bandwidth bw within the interval $[t_{start}, t_{start} + t_d]$. Due to the space limitation, details about how to implement such a function are not described here. A suitable algorithm based on Dijkstra's shortest path algorithm can be found in [2], it can be implemented with complexity $O(E * log(V) * t_d)$, where t_d denotes the duration of the requested transmission. It is obvious, that the search for suitable transmisison interval and rate require more time than fixed requests, however this can be tolerated in the advance reservation environment with the intermediate phase being considerably long (see Figure 2).

Admission control for malleable reservations can be performed in polynomial time. The algorithm described in the previous section requires at most $d_{max} - d_{min}$ cycles of the outermost loop (line 1), $t_{max} - d_{min} - t_{min}$ cycles of the loop in line 3, and an algorithm for implementing the function `find_path` has the complexity $O(E * log(V) * d_{max})$. Hence, the complexity of the admission control algorithm is

$$\begin{aligned} O\big((d_{max} - d_{min}) \quad &* \quad (t_{max} - d_{min} - t_{min}) * \\ E \quad &* \quad log(V) * d_{max}\big) \end{aligned}$$

The basic algorithm performs a scan across the interval $[t_{min}, t_{max} - t_d]$ for each duration $t_d \in [d_{min}, d_{max}]$. We implemented four variants of the algorithm, distinguished by the way this scan is actually implemented. These algorithms are defined as follows:

- **Max/Start**: This variant starts the scan with duration $t_d = d_{max}$ at slot $t_{start} = t_{min}$.
- **Max/End**: This variant starts the scan with $t_d = d_{max}$ at slot $t_{start} = t_{max} - t_d$.
- **Min/Start**: This variant starts the scan with $t_d = d_{min}$ at slot $t_{start} = t_{min}$.
- **Min/End**: This variant starts the scan with $t_d = d_{min}$ at slot $t_{start} = t_{max} - t_d$.

The strategies starting the scan at slot t_{min} are expected to perform better than the other since they avoid heaping up reservations at later time which leads to the gaps as shown in Section 4.3. The initial duration used by the algorithms define their ability to "close gaps", i.e. to fill the space between two peaks. The expectation is, that starting with the shortest possible duration d_{min} is the best approach to fill the gaps since these are too narrow for fixed reservations to fit in and therefore most successfully improves the performance. This can be verified by the results presented in the following section.

5.3 Evaluation

In this section the results of the simulations are presented, showing the performance of the network could be considerably increased with the definition of malleable reservations.

The results were generated with 100 % of the reservations made in advance. A variable percentage of the requests was defined as malleable. Those requests were generated with "preferred" start / stop times and bandwidth, however the network management was allowed to change these parameters. In order to reflect a realistic scenario, we applied some restrictions to the malleable reservations: the duration was allowed to differ at most 50 % from the originally defined duration. The earliest start time t_{min} and the latest stop time t_{max} were at most 20 % of the original duration earlier and later, respectively. This means a request with a given duration of 10 slots was allowed to commence 2 slots earlier than originally specified.

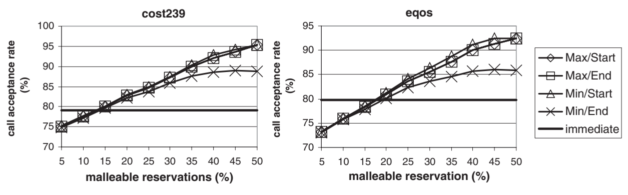

Figure 9. Call acceptance rate using varying percentage of malleable reservations

In Figure 9, the call acceptance rates for the two network topologies are outlined. The upper black line in the diagrams denotes the call acceptance rate when using only immediate reservations (see Figure 4). Not surprisingly, the amount of successfully admitted requests increases with rising percentage of malleable reservations. The strategies Max/Start, Max/End and Min/Start achieve similar results with slight advantages of Min/Start when the percentage of malleable reservations rises. This holds also for the bandwidth blocking rate in Figure 10, where these two strategies achieve the best results. The bandwidth blocking rate can be significantly reduced using malleable reservations.

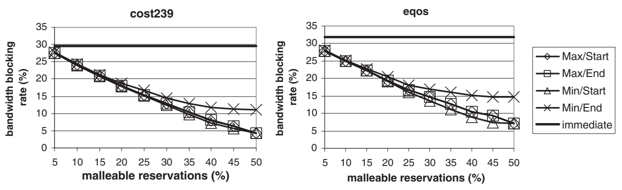

Figure 10. Bandwidth blocking rate using malleable reservations

The reason for the poor performance of Min/End is that this is more likely to produce additional gaps (see Section 4.3) because reservation are placed at the latest possible position. This can lead even to a significantly reduced performance with increasing amount of malleable reservations as shown by these results. The same does not hold for the Max/End strategy because it starts the scan with maximal duration and hence behaves similar to the Max/Start strategy at the beginning of the scan.

The figures previously presented show, that malleable reservations are an opportunity to improve the performance of a network. This makes malleable reservations an interesting opportunity for network operators to increase the benefit of their networks. The following figures illustrate that clients also benefit from defining reservations as malleable.

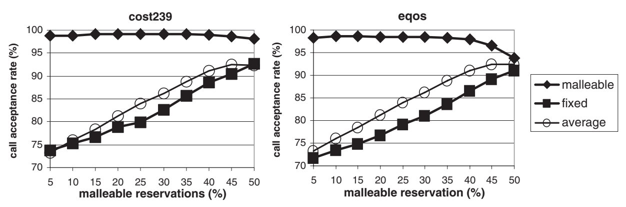

Figure 11. Call acceptance rate of malleable and fixed reservations

The call acceptance rates are depicted in Figure 11, independently for malleable and fixed reservations. It can be observed that malleable reservations have a considerably higher admission probability than fixed reservations, in particular the call acceptance rate of malleable reservations reaches nearly 100 % with only a small drop with rising percentage of malleable reservations.

6 Architecture

In this section the software architecture and the different components of the management system are described that allow to implement malleable reservations in computer networks. The management software can be integrated in a grid toolkit to perform the advance reservations of bandwidth.

Advance reservation services require to store a considerable amount of information which must be accessed during admission control. Hence storing this information in one place, i.e., the bandwidth broker, is therefore preferable over a totally distributed approach such as RSVP. Furthermore, this approach significantly simplifies the implementation of the additional services and optimizations. The bandwidth broker presented here was implemented on Linux and used in a testbed consisting of Linux routers with MPLS capabilities.

6.1 Network

The network architecture using a bandwidth broker on top of MPLS does not vary from others in the field of immediate reservations and therefore is only briefly discussed, concentrating on the topics requiring different solutions in the advance reservation environment.

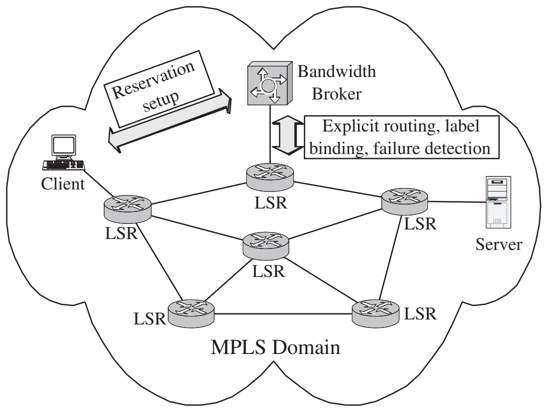

Figure 12. Network architecture with bandwidth broker on top of MPLS aware network infrastructure. Routers (LSRs) exchange label information with each other in order to bind flows to labels/paths.

The underlying network infrastructure is based on MPLS [11] (see Figure 12). Each label switching router (LSR) in the domain is capable of the traffic engineering functionality of MPLS which allows to explicitly define routes for flows. This can be done with arbitrary granularity which means, it is possible to aggregate flows and treat them as one macro flow but also set up paths individually for each flow (micro flow). This guarantees to control the network at any time which is essential for the implementation of the basic advance reservation service.

In order to establish routes between two network nodes, signaling protocols exists such as CR-LDP [9] and RSVP-TE [1]. The signaling is initiated by the bandwidth broker. The setup of routes can be either done each time a flow starts or at start time of the bandwidth broker using the approach described in [2] for the precomputation of routes. This strategy computes a set of k alternative routes for each pair of end nodes. These routes can be set up at the beginning which saves time for setting up routes for each flow at its start time and instead only requires to bind flows to labels, i.e., paths.

6.2 Bandwidth Broker

The bandwidth broker implements admission control as described in Section 5.2. The basic task of the bandwidth broker is to to check whether sufficient resources are available to satisfy a given request, and to communicate with the network components as described in Section 6.1.

The software architecture of the bandwidth broker is depicted in Figure 13. Requests are submitted to the broker using the user interface which can be implemented in sev-

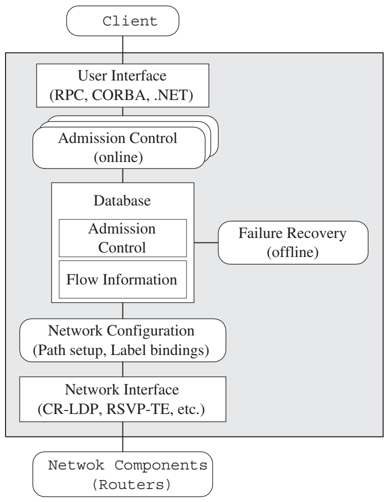

Figure 13. Software architecture of the bandwidth broker

eral ways, e.g., RPCs, Java RMI, CORBA, or .NET remoting. Currently, a CORBA and a .NET interface exist.

Client's requests are processed by one or more admission control processes which use the information stored in the database. This process implements the services and strategies discussed in the previous sections, such as routing or scheduling malleable requests. Once the admission decision is made, the database is updated and the response sent to the client. In case, feedback may be delayed, the request is handed to the offline optimization. In case, no decision is made in time, such request is rejected.

The database is organized as two individual parts. The admission control database keeps information about static parameters such as the network topology, the available bandwidth on each link, and the utilization of the links during the book-ahead period. Furthermore, the link status (UP/DOWN) is stored for each link. The utilization of the network links (see Figure 1) is stored using data structures as described and examined in [4]. In that paper, two data structures - arrays and a specially designed tree - were examined and compared in terms of admission speed and memory efficiency. The result was that arrays are superior in the given environments concerning both metrics, and therefore arrays are used in the bandwidth broker described here.

A second database is required to store information about admitted flows (source and destination node, start and stop time, and path). This database is queried periodically (once per slot granularity) by the a process responsible for the configuration of the network, i.e., the flow-to-label binding and - if required - the explicit routing, i.e., the path setup. Furthermore this process collects information from the network about link failures and updates the respective data in the admission control database. A process is responsible for dealing with those link failures using the strategies described in [3].

6.3 Failure Recovery

An important task of a network management system is to deal with link failures. Strategies to be implemented within a bandwidth broker for advance reservations were described in [3]. The paper proposes selecting the route for a given request from a set of precomputed paths between each two end nodes as discussed in [2]. In case a failure occurs on any link of a path, the flows affected by the failure are ordered, e.g., by their request time, and in this order switched - if possible - to one of the other paths from the set which does not contain the broken link. Furthermore, not only the flows being active at the time the failure occurs are rerouted but also those that are expected to start within the downtime of the broken link.

Strategies as described previously must also be implemented in the bandwidth broker. However, the priority of the failure recovery module should be higher than of the other modules, i.e., admission control and optimization, in order to handle failures as fast as possible and to recover as many affected flows as possible. In fact, it is conceivable that failure recovery completely blocks the bandwidth broker for further requests in order to assure that the maximal amount of affected flows can be switched to alternative paths.

The failure recovery functionality is implemented as an independent module in the bandwidth broker (see Figure 13).

7 Conclusion

In this document, the performance of advance reservations in computer networks was examined and a management software was presented which provides an advance reservation service, as required in particular by grid computing environments.

The performance issue in such environments has so far not received much attention. We examined this aspect in our paper, especially the question how to use the properties of advance reservations, i.e., the availability of book-ahead information, to improve the network performance. The malleable reservation service proposed here goes beyond what has been considered so far in the field of advance reservations in computer networks and it was shown to be successful.

The architecture presented in this paper provides an ideal platform for implementing the different services and optimization techniques that compose the strengths of advance reservations using the available information about future network utilization. Due to the wide availability of MPLS, changes to the underlying network infrastructure are not required and thus the system can be easily integrated in to existing grid toolkits.

Future work deals with scalability issues, i.e., the malleable reservations across several network domains with multiple brokers involved.

References

[1] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP Tunnels. ftp://ftp.isi.edu/in-notes/rfc3209.txt, December 2001. RFC 3209.

[2] L.-O. Burchard. Source Routing Algorithms for Advance Reservation Mechanisms. Technical Report 2003-2, Technische Universitaet Berlin, http://kbs.cs.tu-berlin.de/publications/res_mgnt/tr-2003-02.pdf, February 2003. ISSN 1436-9915.

[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] L.-O. Burchard and H.-U. Heiss. Performance Evaluation of Data Structures for Admission Control in Bandwidth Brokers. In *2002 Intl. Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), San Diego, USA*, pages 652–659. Society for Modeling and Simulation International, 2002.

[5] L.-O. Burchard and R. Lüling. An Architecture for a Scalable Video-on-Demand Server Network with Quality-of-Service Guarantees. In *5th Intl. Workshop on Distributed Multimedia Systems and Applications (IDMS)*, volume 1905 of *Lecture Notes in Computer Science (LNCS)*, pages 132–143. Springer, 2000.

[6] D. Ferrari, A. Gupta, and G. Ventre. Distributed Advance Reservation of Real-Time Connections. 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*, pages 16–27, 1995.

[7] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *7th International Workshop on Quality of Service (IWQoS), London, UK*, pages 27–36, 1999.

[8] R. Guerin and A. Orda. Networks with Advance Reservations: The Routing Perspective. In *Proceedings of IEEE INFOCOM, Tel Aviv, Israel*, pages 118–127, 2000.

[9] Jamoussi, B., ed. Constraint-Based LSP Setup using LDP. ftp://ftp.isi.edu/in-notes/rfc3212.txt, January 2002. RFC 3212.

[10] W. Reinhardt. Advance Resource Reservation and its impact on Reservation Protocols. In *Broadband Islands '95, Dublin, Ireland*, September 1995.

[11] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. ftp://ftp.isi.edu/in-notes/rfc3031.txt, January 2001. RFC 3031.

[12] O. Schelen and S. Pink. An Agent-based Architecture for Advance Reservations. In *22nd IEEE Conference on Local Computer Networks (LCN), Minneapolis, USA*, pages 451–459, 1997.

[13] A. Schill, F. Breiter, and S. Kuhn. Design and Evaluation of an Advance Reservation Protocol on Top of RSVP. In *4th International Conference on Broadband Communications, Montreal, Canada*, pages 430–442, 1998.

[14] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal of Selected Areas in Communications*, 16(7):1228–1234, September 1996.

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

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