Special Issue of The Journal of Multimedia Tools and Applications on
"The State of The Art in Multimedia"
May 1997 (Vol. 4, No. 3)

Concepts for Resource Reservation in Advance

Lars C. Wolf, Ralf Steinmetz

Industrial Process and System Communications
Department of Electrical Engineering and Information Technology
Technical University of Darmstadt*
Merckstr. 25 • D-64283 Darmstadt • Germany
{Lars.Wolf, Ralf.Steinmetz}@kom.th-darmstadt.de

Abstract: Resource management offers Quality-of-Service reliability for time-critical continuous-media applications. Currently, existing resource management systems in the Internet and ATM domain only provide means to reserve resources starting with the reservation attempt and lasting for an unspecified duration. However, for several applications such as video conferencing, the ability to reserve the required resources in advance is of great advantage. This paper outlines a new model for resource reservation in advance. We identify and discuss issues to be resolved for allowing resource reservation in advance. We show how the resource reservation in advance scheme can be embedded in a general architecture and describe the design and implementation of a resource management system providing reservation in advance functionality.

1 Introduction

Computer systems used for continuous media processing must cope with streams having data rates of several Mbits/s and must provide timely processing guarantees. For instance, an end-system shall synchronize audio and video streams up to a granularity of about 80 ms [16]. Since available system resources are not abundant, applications have to be 'protected' such that they have access to the required resources in time. Otherwise the user will notice a glitch or drop in the presentation quality. Hence, means to manage the available system resources are necessary.

Resource management provides a way to offer applications reliability with respect to *Quality-of-Service* (QoS) [21]. A resource management system controls the access to scarce system resources needed for audio and video data processing. It checks whether additional service requests can be satisfied, and if yes, the required resources are reserved for that application, else, the request is rejected. Sophisticated systems will allow for a negotiation according to the available capacities and constraints (e.g., by tariffs).

1.1 Requirements of Application Scenarios

Today existing resource management systems, for instance, HeiRAT [18, 19], QoS Broker [12], Tenet [8, 1], offer functions which only allow to reserve resources for a time interval which starts with the reservation attempt and which lasts for an unspecified time.

For several application scenarios this model of immediate reservations is not appropriate. Consider, for instance, a virtual meeting room (conferencing) scenario supported by multimedia systems. Traditionally, a meeting will be scheduled for a specific time at a well defined location (room). To be sure that the respective room will be available at the scheduled time, a

*. Part of this work was done while the authors were with IBM's European Networking Center, Germany.
This work is sponsored in part by: Volkswagen-Stiftung, D-30519 Hannover, Germany.

reservation entry, in some form of a meeting room calendar, is written *before* the meeting starts. The time between the reservation and the meeting itself can vary from short intervals, e.g., half an hour or a few hours, to very long periods, e.g., months. In addition to 'one time events', meetings such as project meetings occur periodically. To support these 'virtual meeting room' scenarios the resource reservation system must offer mechanisms to reserve in advance the resources needed for the conference, i.e., certain capacities of networks, routers, and end-system resources.

Resource Reservation in Advance (ReRA) is not only needed for conferencing but for other scenarios such as video-on-demand as well. This resembles a video rental scenario where a user 'orders' a video for a specific time: for the video-on-demand system it means that the resources necessary to retrieve, transfer and present the video have to be reserved in advance, i.e., video server, network, router, and end-system resources. Further application areas can also be found outside of typical multimedia applications, e.g., within manufacturing process control systems (where time-critical data must be processed and transmitted) or any kind of remote surgery in medicine. The need for ReRA has also been expressed by Degermark, Köhler, Pink and Schelén in [5, page 4] as:

> "Where resources are plentiful, not even immediate reservations may be necessary, but where resources are scarce enough to justify reservations at all, it makes sense to be able to make them in advance."

1.2 Contents of this Paper

We discuss a model for ReRA, identify the issues to be resolved and describe the design and implementation of a resource management system offering ReRA functionality. The paper is organized as follows: Section 2 provides a description of a common resource management scheme and an according architecture. Section 3 introduces the notion of ReRA, the used model, and characteristics of ReRA. Section 4 presents the architecture of a ReRA system and some aspects of the prototype implementation. Related work is discussed in Section 5 before we conclude the paper.

2 Resource Management

In the following we describe the main issues of resource management relevant to the ReRA scheme. For a more detailed view please refer to, for instance, [19, 12 and 21].

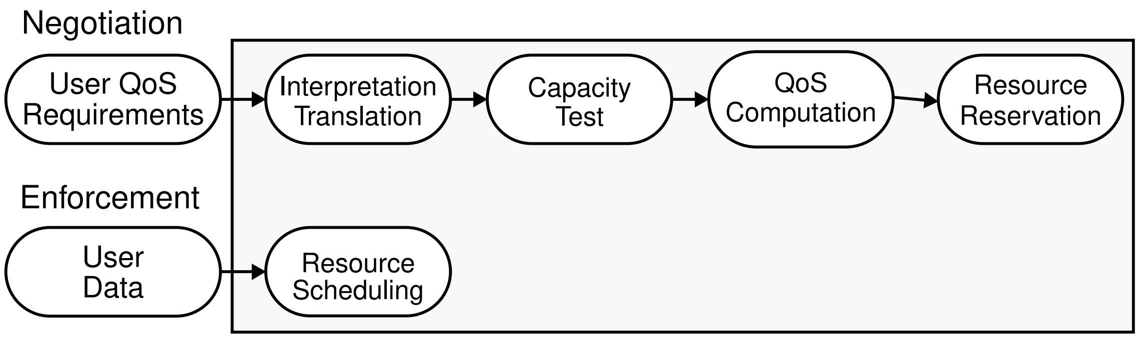

Figure 1: Resource Management Phases.

Any resource management component on each system which is part of an application must provide certain functionality for each "active" resource (i.e., CPU, network adapter):

- *Interpretation and translation* of the application specified QoS in metrics applicable to the affected resources.
- *Capacity test* to check whether the available resource capacity (taking the existing reservations into account) is sufficient to handle the new request.
- *QoS computation* to calculate the possible performance the resource can provide for the new stream.
- *Resource reservation* to reserve the required resource capacities.
- *Resource scheduling* to perform the scheduling of the resource during data processing such that the QoS guarantees are satisfied.

Figure 1 illustrates how this functionality interact at two resource management phases. In the set-up phase (also called 'QoS negotiation') applications specify their QoS requirements (e.g., throughput and delay). These parameters are used for capacity test and QoS computation which finally results either in resource reservation or in rejection of the reservation attempt if the QoS cannot be met due to a lack of resources. After the negotiation phase has been successfully completed, in the data transmission phase, the resources used to process the user data are scheduled with respect to the reserved resources (also called 'QoS enforcement').

In a ReRA system, the negotiation phase is not in close vicinity to the enforcement phase and the resources are not reserved for immediate but for delayed use.

The resulting states for a stream are `idle`, `negotiate`, `established` and `active` (see Figure 2). During the set up of a stream, it changes due to a `request` from `idle` to `negotiate`. If the set up was successful (`accept`ed by the resource management system) the stream is `established` which is in a non-ReRA system identically with `active` (dashed box). In a ReRA System, both states must be distinguished since negotiation and usage can be potentially very distant.

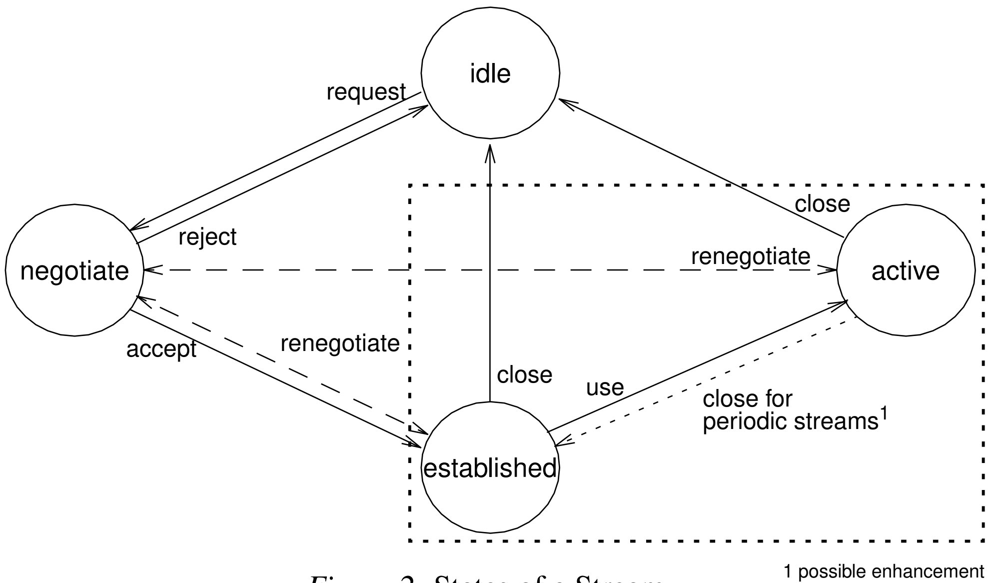

Figure 2: States of a Stream.

To achieve QoS provision for a distributed multimedia application, resource management is applied to all resources on the transmission and processing path, from the sending host via gateways or any other computers and networks to the receiving host. Resource reservation protocols such as ST-II [17] and its more recent version ST2+ [6], and RSVP [22] offer the functionality for QoS provision in distributed systems.

The local resource management system contains among others components for QoS interpretation and translation, schedulability tests and the according scheduling mechanisms, databases for available and reserved resources, monitor and regulator mechanisms to detect and potentially enforce the characteristics of data streams. A simplified architecture is shown in Figure 3.

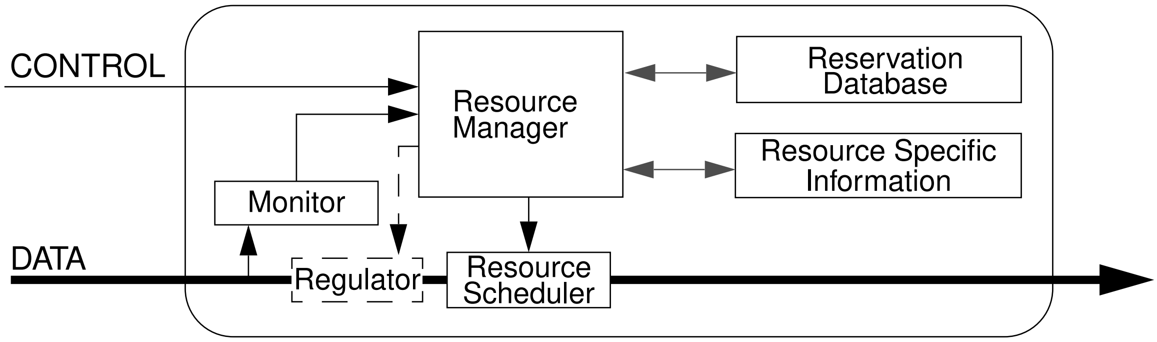

Figure 3: Components of a Resource Management System.

3 Resource Reservation in Advance Model and Issues

This section introduces the notion of ReRA, presents its basic ReRA model, and discusses issues to be addresses by ReRA systems.

3.1 Classification of Reservation Types

To distinguish ReRA schemes from other reservation schemes, e.g., existing reservation techniques, we classify reservations based on two key factors:

- whether the resources are exploited at reservation time, and
- whether the reservation duration is known at reservation time.

The most stringent use of resource management is in the domain of process and control systems including embedded real-time systems. There, resources are reserved for the whole active phase of such systems, i.e. for the lifetime. Changes can only be done at the initialization phase (and not at the actual run-time phase). Therefore we characterize such approaches as “static” (see Figure 4) opposed to the dynamic approaches discussed in the remaining of this paper.

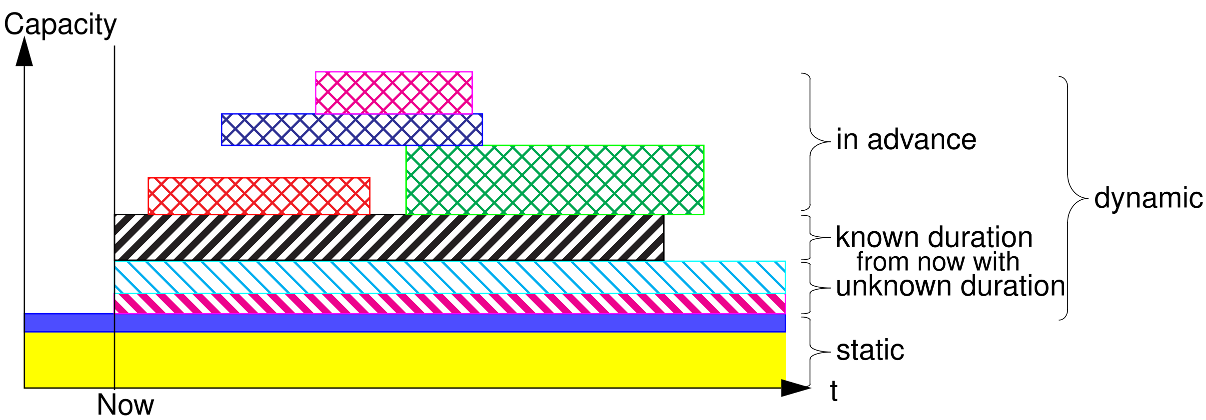

Figure 4: Reservation Types.

Traditional resource management systems (non-ReRA) assume that the resources are immediately used after they have been successfully reserved and no assumptions are made on the duration of the reservations. A ReRA scheme, on the contrary, is characterized by deferred resource usage and reservations of known duration (which might possibly be enlarged).

In case of immediate usage and known duration, both schemes can be realized. We clarify this point at the end of this section, after introducing the ReRA model.

This leads to the simple matrix presented in Table 1:

Table 1: Classification of Reservation Schemes.

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">Duration of Reservation</td></tr>
<tr><td>Known</td><td>Unknown</td></tr>
<tr><td rowspan="2">Dynamic Usage</td><td>Deferred</td><td colspan="2">ReRA</td></tr>
<tr><td>Immediate</td><td></td><td>Traditional Approaches</td></tr>
<tr><td colspan="2">Static Usage</td><td colspan="2">Unlimited Duration</td></tr>
</table>

3.2 Basic Model

To provide an appropriate model for ReRA, we start from the common reservation scenarios of everyday life. In such scenarios, appropriate actions are required as part of the reservation, e.g., we have to specify at what time and for how many persons we intend to reserve. Here, we introduce a simple model to define these actions and regulate the interaction between the reservation requestor (i.e., the client application itself or a ReRA agent acting on behalf of the application, cf. Section 4) and the service provider (e.g., network and server applications). The model is shown in Figure 5.

The ReRa scheme consists of two parts:

- resource reservation in advance
- usage of reserved resources

In the first part of the ReRA scheme, the client issues a REQUEST and it specifies the nature of its request by indicating how much of the resource capacities will have to be reserved for its application, i.e., it gives a *workload specification*. It also specifies the points in time that define *beginning* and *duration* of the reservation. The service provider may then CONFIRM the reservation. As part of this confirmation, it possibly provides the client with a *reservation identifier* for later client identification. This terminates the first part of the ReRA scheme.

The second phase begins shortly before the client intends to exploit its reservation. The client contacts the service provider to DEMAND the previously reserved resources. It may be requested to show some form of identification, which the service provider will ACCEPT. After receiving this acceptation, the client eventually exploits its reservation by making USE of the reserved resources.

While a session is established, the participants may like to either finish earlier (than previously reserved) or they may like to extend the time. The drop of the session before the actual reservation time expires is what usually will happen, it is shown by the close primitives in

Figure 5. The extension request is shown as a continue primitive which again must be confirmed by the whole set of involved resources.

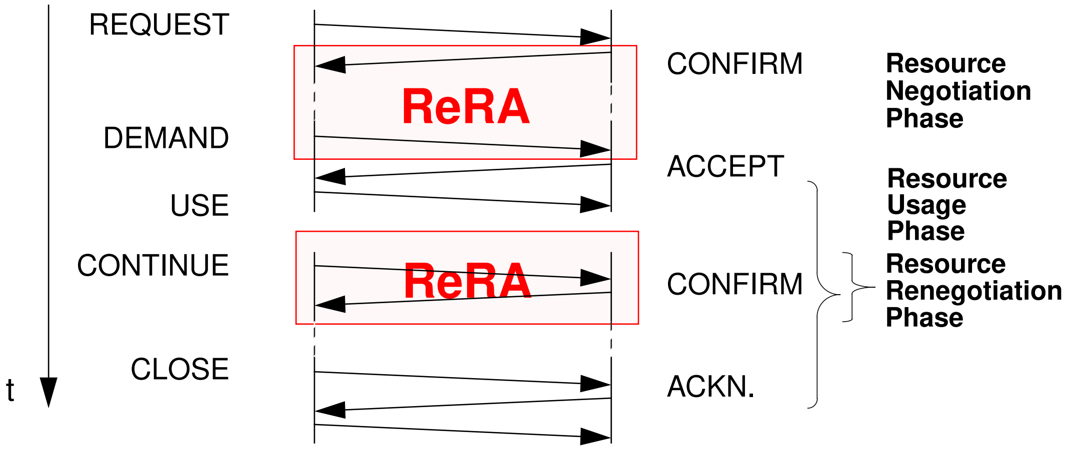

Figure 5: Reservation in Advance Primitives.

It is possible to further simplify this scheme by eliminating DEMAND and ACCEPT. In this case, the client attempts directly to make use of the allocated resources and client identification can be associated with USE. However, we feel that the scheme described in Figure 5 is more convenient because it provides for the management system the ability to prepare the resource usage phase and generally allows for higher flexibility. For instance, it is often necessary to change reservations at the very last moment. A common example is a couple of unexpected guests for dinner making a larger table necessary. With the DEMAND and ACCEPT scheme, DEMAND can be used to adjust reservations appropriately when possible. Also, an explicit ACCEPT from the service provider is desirable because it informs the client that everything is set so that its requirements can be met.

3.3 Timing

In order to appropriately define a ReRA system, it is important to analyze the temporal relationships among the events. Consider the events in Figure 6.

In our view of a ReRA system, we assume that the distance between t_{req} and t_{conf} is short, about the order of delay tolerated by Remote Procedure Calls (RPC). The same holds for t_{dmd}, t_{acc} and t_{begin}. On the contrary, t_{req} and t_{begin} are possibly very distant, possibly in the order of weeks or months. Let us call I the time interval between resource reservation and exploitation:

$$I = t_{begin} - t_{req}$$

When I is too small, making a ReRA reservation is pointless and a normal reservation scheme can be adopted. A ReRA system may define a value for I, say I_{min}, such that requests with:

$$t_{begin} - t_{req} < I_{min}$$

are rejected because the overhead induced by the management of resources might otherwise be too large. In the same way, an I_{max} value for I can be defined to prevent applications to request their reservation long ahead of time, e.g., to prevent storing too much reservation state. These definitions help clarify Table 1 when both non-ReRA and ReRA are possible, the I_{min} value can be used to decide which of the two schemes to adopt. However, if limits for I are defined it must be considered that in the 'real' world we have the possibility to specify exceptions, e.g., if hotel rooms for a large event can be reserved long time in advance, similarly, it might be necessary to be able to reserve network and other system resources for large events already long time in advance.

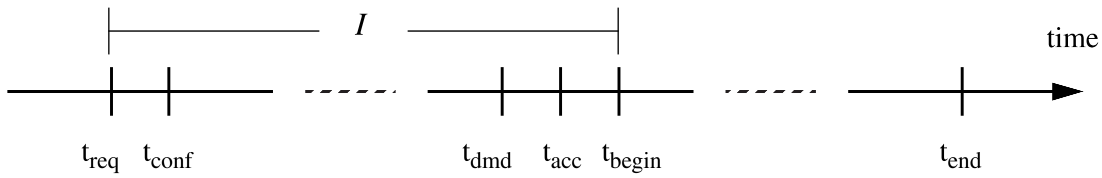

Figure 6: ReRA Model Temporal Sequence.

Instead of using single values for the events, the use of intervals and a target value within this interval is more general. For instance, instead of the single value t_{req}, the interval [$t_{req,\ earliest}$, $t_{req,\ latest}$] and the according target value $t_{req,\ target}$ might be used. If a certain application will be run periodically, e.g., the video conference of the weekly project meeting, it is useful to specify the reservation already as periodically.

All specified time values must be unambiguous within all components participating in the provision of a distributed and ReRA supported application. Hence, absolute time values, based on synchronized and coordinated clocks must be used. The granularity of specified times might be system dependent and influence the overhead incurred by the ReRA system. As for 'real-world' reservations, we believe that introducing a certain granularity, e.g., a virtual conferencing room is only reservable for one or several slices of 15 minutes, would be acceptable to users. If necessary, a distinction among 'close' and 'far away' events may be added, e.g., for events starting soon the granularity might be set to one minute, for events far in future it might be 30 minutes.

3.4 Reservation Duration

ReRA schemes require that the applications reserve resources over a certain time interval. The problem is, it is difficult to predict in advance how long some applications may need their reservations. In a video-on-demand system, it is usually possible to foresee the duration of a movie. Still, the user may increase this duration by pausing playout or even by stopping and rewinding to watch his favorite sequence a second time. In the same way, meetings take often longer than expected. Note that also shorter durations may be induced, e.g., by skipping through movie sequences or by rapidly adjourning a meeting.

When the actual duration does not correspond to the reservation, several issues arise:

- if the *duration is shorter*, exceeding resources should be freed and made available for other applications. In this case, resources are more likely to be made available for immediate use and for traditional reservation requests than for new ReRA requests, because of the short notice (which is likely < I_{min}).
- if the *duration is longer*, the system may or may not have a sufficient amount of resources to serve the application with its needed QoS. If enough resources are available, one possibility is not to interrupt the service and to provide the application with the means to extend its previous reservation. If insufficient resources are available, the system may still attempt to serve the application on a best-effort basis with a degradation in the QoS.

Means to extend a previous reservation are desirable for a ReRA system, i.e., in addition to the primitives discussed in Section 3.2 a CONTINUE primitive to enlarge an already established reservation is necessary.

Sometimes, the delay can be foreseen, e.g., it becomes clear that the meeting will take longer than expected. In such cases, it may be possible to extend in advance, i.e., before it expires, a previous reservation. This will only be successful if sufficient resources are available, e.g., if no other reservation overlaps with the extended reservation. For the prolongation of the reservation, we differentiate two alternatives, (i) the management system informs the application/user that the reservation will expire and queries whether the reservation should be extended or (ii) the application has complete responsibility about the reservation state and must take appropriate action to lengthen the reservation.

The ability to extent reservations encourages applications not to book resources over too long time intervals in order to be guaranteed against unpredicted longer durations. In a cost-based ReRA system, this can also be imposed by adequate payment policies of the associated reservation costs.

If it is known before the beginning of usage that the needed reservation duration is different to the originally specified length, the DEMAND mechanism can be used to adapt the reservation to the required duration. If the duration shall be shortened, the reservation requestor might be charged for preventing other reservations. For prolongation, the necessary resources might be unavailable, however, due to the earlier request, the risk is lower than during the usage phase of a reservation.

3.5 States

The state diagram for non-ReRA streams as shown in Figure 2 must be extended to allow for the reservation of resources in advance. That figure distinguishes already between `established` and `active` which are identical in a non-ReRA system. Yet, this simple extension falls short, e.g., it does not provide the ability to handle failure situations which can occur between the reservation and the use of the resources which can be very distant in a ReRA system. Yet, in the time between reservation and use, resource characteristics and availability might change. Therefore, additional states must be introduced as shown in Figure 7.

The regular 'state cycle' consisting of `idle`, `negotiate`, `established`, `active`, and `ending` is basically identically with the one described above. However, between `established` and `active` time can pass and, hence, resource capacities might change in such a way that the negotiated QoS can no longer be guaranteed. In that case, the stream is still established but moved into the `can't be met` state; if later the capacity becomes sufficient again its state is changed back to `established`. Otherwise, the application may renegotiate its resource requirements or close the stream. Serving an 'established' stream with insufficient resources is not considered as useful because then a stream without any resource reservation can be used.

The timely distinction between the reservation and the use of resources implies that the begin of the usage phase (marked by the `use` primitive) can occur before, at or after the scheduled begin date. Therefore, three cases have to be distinguished:

- The use request is given too early. The stream is marked as 'in-use' and put into the state `early` but not yet scheduled with real-time priority. Hence, it is served in a best-effort mode without any QoS guarantee. When the reserved time is reached (`start`) the stream changes into the `active` state and it is scheduled with real-time priority in order to achieve the negotiated QoS.

- The use request occurs exactly at the specified time. Then the stream changes directly to the `active` state and is scheduled with real-time priority. This case is, depending on the granularity of time measurement, the exception and not the regular case.
- The use request comes within the reserved time interval. From the beginning of the reserved interval until the use request the stream is in the `pending` state from where it changes to the `active` state. If the `use` never occurs (or occurs very late, i.e., after a specified 'reservation holding time') the reservation expires and the stream goes via `expired` into the final `idle` state.

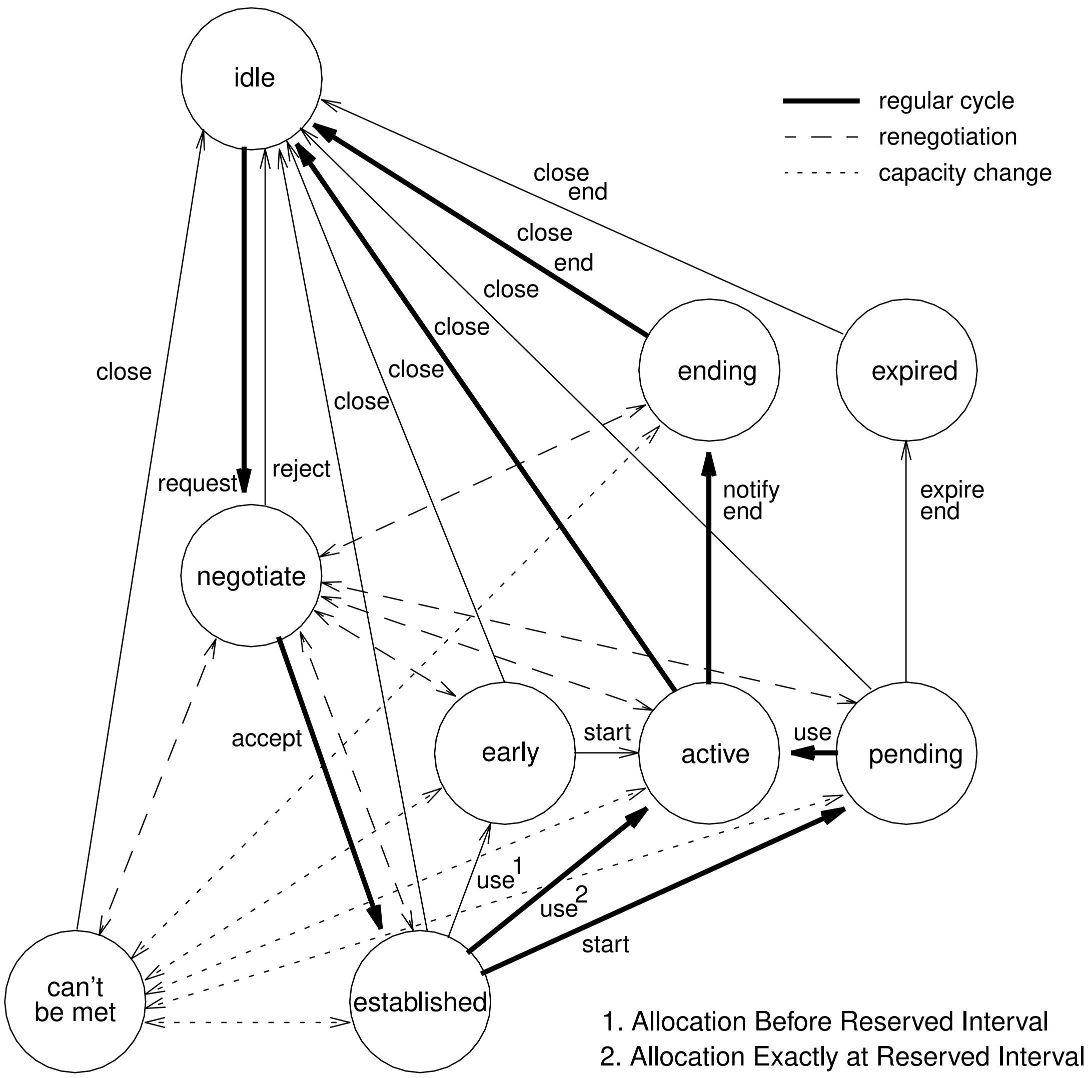

Figure 7: Extended State Diagram for Advance Reservation With Additional Characteristics and Error Conditions.

A certain time before the reservation ends the application can be notified by the system about this coming event (and changes into the `ending` state). Hence, the application can prolong and renegotiate the reservation.

3.6 Distribution of Announcement Information

In addition to the information about stream characteristics which are exchanged via resource reservation protocols such as RSVP and ST-II, information about the date of the stream and even basically the knowledge about its existence time must be distributed as well. Such information is today usually distributed via other means than the one later used for the application, e.g., the invitation to join a multi-user phone conference is given to the potential participants by contacting each person independently via a point-to-point phone call.

Considering distributed multimedia applications, two different scenarios can be distinguished:

- the sender of the data stream has complete knowledge about the set of receivers, i.e., their identities,
- the sender has no knowledge about the set of receivers, i.e., the sender knows neither identity of receivers nor even whether anyone is listening.

The former scenario occurs in point-to-point communication or in multicast communication using a sender oriented communication setup, this is the case, for instance, in ST-II. The latter scenario is used in multicast communication where the receivers are not specified by the sender and is typical for IP multicast communication.

In both cases, the human users who will consume the presentation of the transmitted data must be informed about the intended transmission of the multimedia data and accept or reject the stream. While it would be possible, in the first scenario, to perform the announcement phase together with the reservation attempt, it should be avoided. The reason is the severe drawback that until the user decided about the delivery of the stream, the resources on the complete path from the sender to the receiver must be set aside (for the future time frame) for the case that the receiver accepts the stream. However, the reaction of the user might be delayed for an unknown amount of time – the user might even not be at the computer for days or weeks, e.g., during business trips or vacations. Keeping resources reserved for such a long time can lead to rejected reservation attempts for other applications even if finally the user decides not to receive the stream and, therefore, available resources at the time the other applications intended to run. Additionally, it complicates the implementation of the ReRA components and the recovery from system failures. Altogether, it becomes clear that a distinction into the *announcement* of the data stream and ReRA for that stream is necessary.

The information about these announcements can be handled by a 'user agent' which is similar to the user agent of a mail system. It provides the interface for the user to handle resource reservations in advance. An incoming invitation to a multimedia application (to be started sometime in the future) is presented to the user who can acknowledge or reject the invitation. Using this agent, users can also start reservation attempts themselves. The user agent should provide the ability to start automatically the application at the time the data stream has been scheduled, i.e., just before the conference begins.

3.7 Failure Situations

For the handling of failures, we must distinguish when the failure occurs:

- during the negotiation and reservation,
- after the reservation, but before the usage (between REQUEST and USE), or
- during the usage phase.

The first and the last case is not different from failures within traditional reservation-based systems. The second case, however, requires special attention since it was not present in non-ReRA systems.

The reservation state information stored at nodes might be needed for long lasting time periods. State information must be stored in stable storage not only to protect against failures, but also since any node may be restarted between REQUEST and USE also regularly, e.g., for maintenance.

In opposite to failures occurring during data transmission, no client is running when a node notices a failure. The failure itself might, however, not be detected at the failing node but only at a neighbor which has only partial information about the reservation state stored at the node. Means to inform the clients explicitly about the failure situation and whether it can be resolved in time must be provided, respectively the application must be able to query the correctness and availability of the reservation before it starts its usage phase.

Applications which have been written for a scenario where resource reservation is done but which are not aware that they are running within a ReRA system might encounter during their resource reservation usage phase the additional problem that the resources are not available as expected. This can occur if the application is early and wants to use resources before the reserved time interval or if the reservation interval ends and the application continues to use the resources, e.g., if a video conference lasts longer than originally scheduled. In the first case, the application is served on a best-effort basis until the begin time of its reservation, hence, the perceived QoS might be low. In our system, the application is informed about this situation by an according flag in the acknowledgment of the allocation. In the second case, the resource reservation ends and the application is served from that time with a best-effort mode only. Yet, the system offers the ability to inform the application *before* the reservation ends (state `notify` in Figure 7). Then the application can extend the reservation if necessary. This functionality will lead to optimistic reservations, i.e., an application will usually reserve for tight bounds and not necessarily reserve for a surplus of time. Service provider can even foster such a reservation behavior (which is desirable for them) by offering better rates for such reservations.

If an application which reserved resources in advance fails to start and hence does not use the reservation, the service provider will probably nevertheless charge the application (resp. its user) somehow because the resources have not been available for other applications whose reservation attempts might have been rejected. This is similar to reserved but not used hotel rooms, etc. In order to reduce the effect, a time after which reserved resources are made available to other applications is defined. The state `expired` is introduced for this purpose.

3.8 Management of the Resource Pool

It is not possible for all applications to know in advance their duration, e.g., video conferences. For other applications, it is not possible to determine (long enough in advance) when they will run. Therefore, advance reservations and immediate reservations must coexist.

Advance reservations may block immediate reservations if both are served from the same pool of resources (see Figure 8). This blocking can occur if at a later time all resources are reserved by advance reservations. The reason is that since the duration of an immediate reservation is unknown, it must not overlap with any existing advance reservation. Hence, the reservation is rejected despite all needed resources may be available. The same applies to advance reservations. They are rejected if existing reservations (immediate and advance) use all resources, since immediate reservations can last for undefined time.

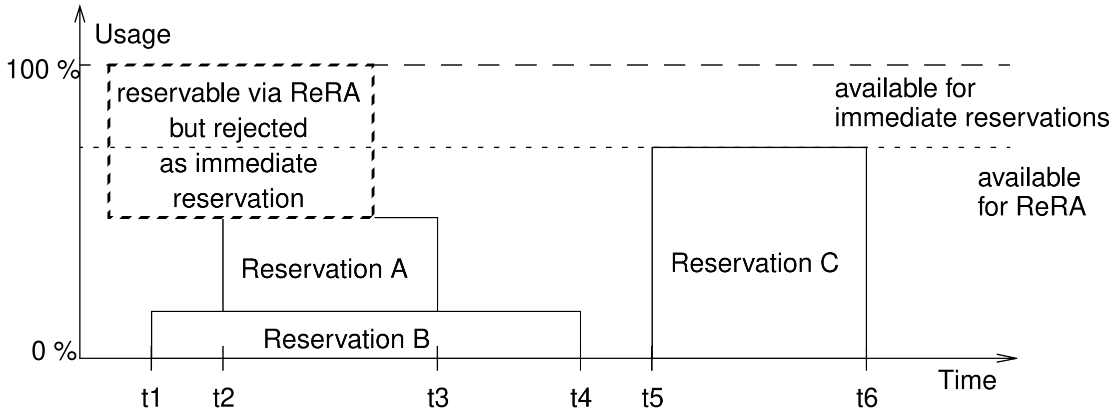

Figure 8: Partitioning of Resources.

This blocking problem can only be solved by splitting the available resources among the two reservation styles. A simple scheme would be to use fixed-sized partitions – one part is used for immediate, the other for advance reservations. However, it is nearly impossible to find partition sizes which satisfy the reservation needs over a longer interval for varying requirements. Thus, partitions with floating boundaries must be used, to avoid that one partition becomes empty (and the other contains all resources) water marks for minimum values are needed.

In Figure 8, an example for the use of resources is illustrated. For the intervals $t_1 - t_4$ and $t_5 - t_6$ advance reservations have been set. As can be seen, advance and immediate reservations block each other. The dashed part could be reserved as advance reservation but it cannot be reserved as immediate reservation because its duration is unknown.

Instead of using only two partitions, one for immediate and one for advance reservations, more partitions for advance reservations can be used to distinguish among short and long-term reservations.

4 Architecture for Resource Reservation in Advance

Now we describe the architecture of our ReRA system and some aspects of its prototypical implementation.

4.1 Management of Reservations

To allow for reservations in advance, the time axis is divided into slices. Within each slice a certain set of reservations exists and there is no change of this set or of the QoS parameters of these reservations, i.e., the reservation state is stable within each slice and changes only at the boundaries (as described in [7]). Thus, the resource management system has a similar view as before: at a certain point in time (in a time slice) a fixed set of reservations with fixed QoS exist corresponding to a fixed resource utilization and free resource capacity. This view changes only if new reservations are established or existing ones end. Therefore, the following components of the resource management system need modification:

- The interface of the resource management system needs in addition to the QoS parameters now also specifications of the time parameters (begin and duration).
- These time values must also be contained in the flow specification distributed via the resource reservation protocols to all affected network nodes.

- The database of existing reservations must represent the time slices. For each time the set of existing resp. reserved streams with their QoS parameters and the free resources must be known.
- The resource management algorithms must take the time parameters into account.
- Additional failure handling mechanisms and means to save state information in permanent storage are necessary.

Furthermore, the reservation protocols must be enhanced. New PDU types to support the additional states and transitions (e.g., USE explained in Section 3.2) and to handle failure situations and notify neighbor nodes about such are needed.

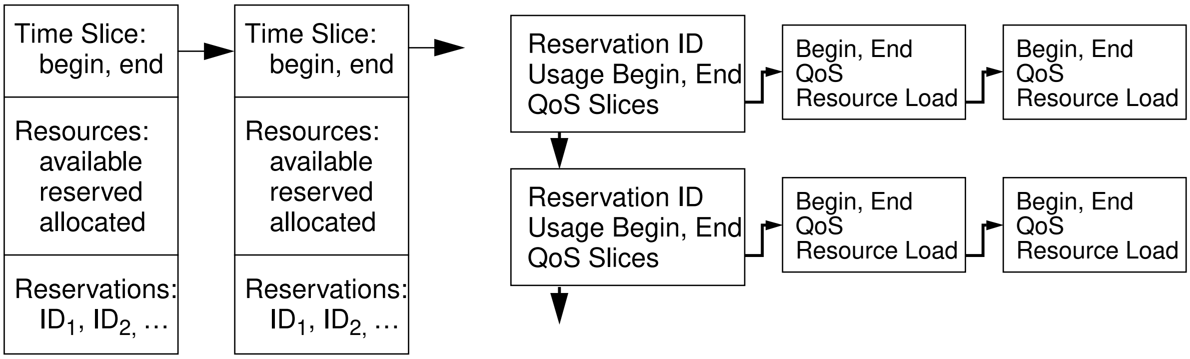

Figure 9: Data Structures for Management of Time Slices (left) and Reservations (right).

Figure 9 illustrates the data structures for the management of time slices (left side) and for reservations (right side). Information for all slices is kept in a list so that splitting a slice or joining adjacent slices is simple. For each slice the information about its beginning and its end, the amount of resources (available, reserved, and allocated) and the identifiers of set reservations are stored. Binding available resources to time slices allows the change of resource capacities over time. By keeping separately track about reserved and allocated resources provides the possibility to distinguish between established, active and not serviceable reservations (cf. state diagram in Section 3.5).

The list for all reservations contains one entry per reservation where a list of times, the (potentially varying) QoS within this time, and the corresponding resource load is stored.

The data structures for the management of the resources hold information about the used scheduling algorithm, the amount of resources to be managed and further parameters affecting the schedulability test (e.g., the maximum packet size for a transport system). This provides for the check whether sufficient resources are available for a new stream.

4.2 Management of Resources Characteristics

The usable capacity of a resource can vary within a long time interval, for instance, due to necessary maintenance work only parts of the full capacity, e.g., in a network, might be available. Therefore, a system component independent of the reservation management should exist which keeps track of the capacities and characteristics of the managed resources.

The time of the reservation of resources does not necessarily coincide with the beginning of the usage phase, hence, the reserving application is in the mean time usually not active and reachable. Thus, in case of changes, another instance must be available which can implement corresponding reactions. This part can be taken over by the reservation management – it is informed about resource capacity changes and checks then whether all active and reserved streams can still be served. If the available resources are not sufficient to serve all these streams, some of the streams must be modified. For active streams, the application can be informed, whereas for reserved but not yet active streams, the application might not be reachable now. It will be informed about the changed situation when it contacts the reservation management, i.e., when it wants to use the reserved resources.

The introduction of components for resource dependent functions, i.e., for capacity determination and for mapping between QoS and resource load, generally provides for independence of the reservation management from the actual resources and the usage scenario of the reservation system. Complete independence is not possible, since several policy decisions implied by the usage scenario must be taken by the reservation management, e.g., the decision which reservations are marked as `can't be met` in case of resource capacity reduction. Further, the workload model has impact on the behavior of the resource and the used scheduling algorithm influences the maximum resource utilization (cf. [11]) and hence the amount of available resources.

4.3 Notification Component

The reservation management should be designed as general as possible and independent of resource and application specific details. In the last subsection this led already to the distinction between management of reservations and resources. To allow for independence from applications and reservation protocols, an additional component is introduced. This notification component accepts 'sign in' and 'sign out' of an application and hence knows whether the initiator of a reservation can be reached. Further, it informs an application in case that the state of one of its reserved streams changes, e.g., time controlled or due to resource changes. Whether this notification of the application by the notification component occurs immediately or delayed depends on the reachability of the application which is known by the notification component due to the 'sign in' and 'sign out' operations.

Some applications might not be aware of the notification mechanism, hence, it cannot be mandatory. Applications can specify that they want to be informed about state changes and which state changes should be traced.

4.4 User Agent

A user agent is introduced to allow the user to coordinate its work flow with the use of certain applications. The user agent, similar to a calendar and comparable to a mail user agent, provides the ability for a user to make reservations for a specified time interval and to visualize statistical information about the reservations stored within the resource management system, for example to find out the best time interval for a new reservation.

Another task of the user agent is to control the state of existing reservations: Can a reservation still be served ? Till when will the failure be repaired ? Could the stream be served with lower QoS ? Another useful feature is the detection of the start time of the usage duration and the automatic start of the application which serves also as reminder.

The user agent is additionally the component which receives announcements about events such as conferences from other users. It notifies the user about that event who can then decide to accept the invitation and participate in the event. From the information given inside the announcement, the user agent should be able to deduce an according reservation.

4.5 Announcement System

The user agent integrates the announcement service into the reservation system. The characteristics of the event such as date, duration, required QoS and additional information such as agenda, contact person, participants, etc. are specified via the reservation tool and made available with its help to the other users, e.g., the participants of a conference call or all interested persons in case of public events.

Using their user agents, other users can search in public databases containing event announcements and perform reservations with database provided information for the events which are of interest to them. Such databases can be compared to magazines or announcement boards.

For private events, the number of interested persons is usually limited (confidential information) or the interest is geographically concentrated (local announcements). Therefore, the announcement can be sent to each interested party personally via email.

The architecture of the announcement system is illustrated in Figure 10. The 'announcement & reservation tool' is used to specify the event characteristics. Then the event is either announced in a publically available database or the announcement is packaged into a form and send via email to the participants.

Users search for public events in the databases. Private announcements are taken and filtered from their incoming mailbox and put into a separate box which is checked periodically by the reservation tool. Then the user only has to accept or reject the invitation to join the announced event. In future, the announcement generation and processing might be integrated into calendar and groupware applications.

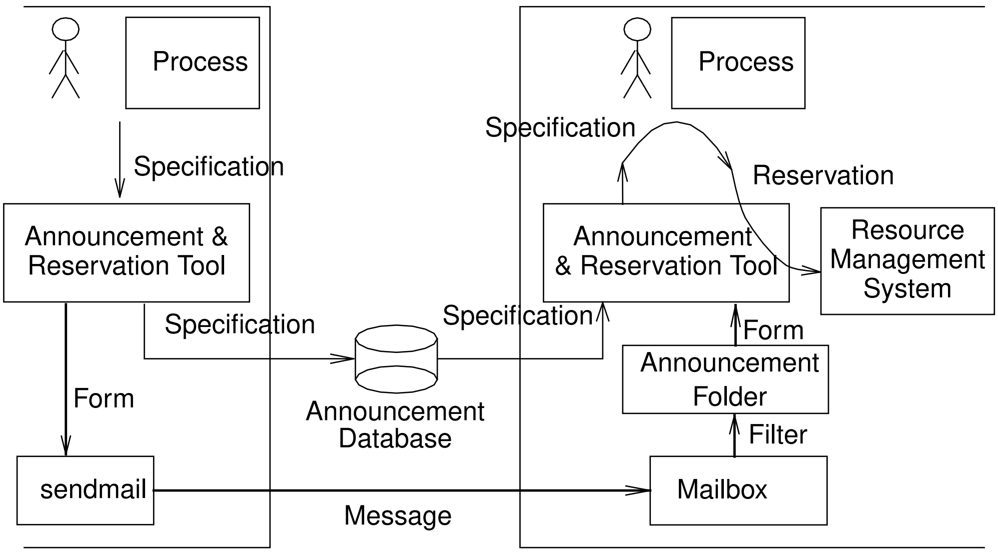

Figure 10: Architecture of the Announcement System.

4.6 Overall Architecture

The described components interact as illustrated in Figure 11. The central part is the reservation management which uses the databases holding the information about the resource characteristics and the usage of resources over time intervals. The resource database keeps information about the resource capacities for every point in time. The reservation database stores for every time interval the information about existing streams and their resource load and also about the available resources in that interval. With this information, the schedulability test can decide whether a new request can be satisfied.

In case of a change of a time interval (a new time slice starts), the resource scheduler gets information about all active streams and their priorities. Based on this, it enforces the QoS by appropriate dispatching.

The mapping between QoS parameters and resource requirements must be done before the schedulability test can decide whether to accept or reject a request. This mapping is done within a resource-specific component which also keeps track about changes of resource capacity and characteristics.

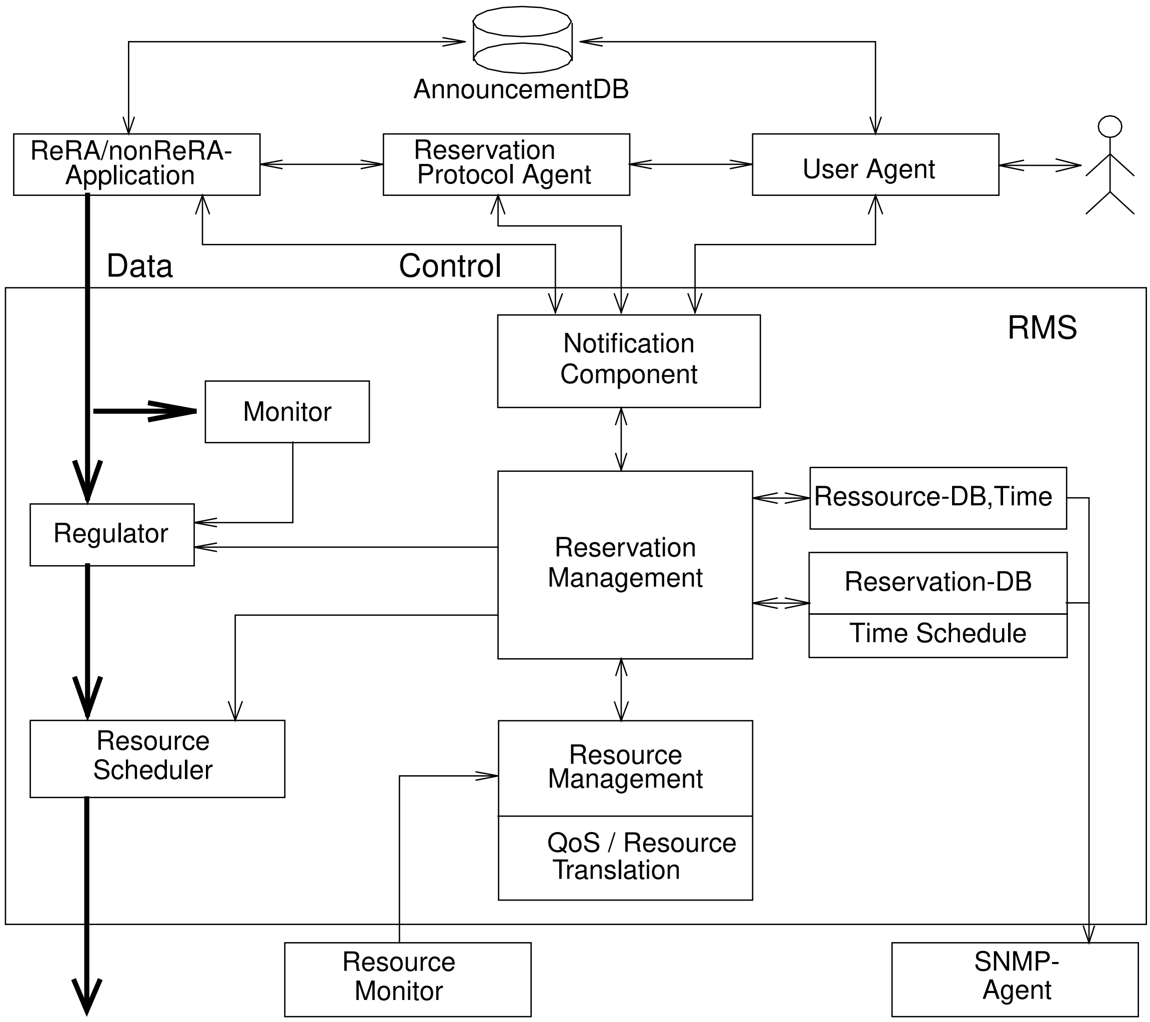

Figure 11: Architecture of a ReRA Resource Management System.

The notification component offers the interface to the using system by accepting the service requests from applications. Resource requests and also requests for statistical information (e.g., state of a reserved stream, available resource capacity within a certain time interval) are forwarded to the reservation management. The resource reservation requests come from a local application, from the protocol engine of a resource reservation protocol, e.g., RSVP or ST-II, or from the user agent.

The user agent is only a ReRA application, reservations performed via the agent cannot be distinguished from applications made by other applications. The difference is outside of the RMS. A 'normal' application reserves resources for a particular work effort and duration, the user reserves in order to have needed resources available *at a certain time* for his task (then means to hand over the reservation to an application are needed).

The user agent offers an interface to the announcement database. It can be used to announce own events and perform queries of announced events. If a user schedules an event such as a conference call, he must inform the other participants about the beginning and duration of the event and must also provide information about the required QoS parameters. Additional information such as topic/agenda, participants, etc. is useful. Some events such as conferences or lectures will be public, others such as conference calls are private. The announcements of public events are stored in public databases where the users can perform the above mentioned queries.

For system administration purposes, a SNMP (or CMIP) agent should be attached to the RMS as indicated by the dashed lines and box on the right side of Figure 11. Administration would also be possible via the regular interface through the notification component, however, because of standardization and security reasons a separate interface is preferable. For instance, any user should only be able to retrieve information about his/her personal reservations and anonymous information about the rest. However, system administrators need access to all reservation information.

4.7 Implementation Decisions

We decided to place no limit on the period of time for reservations in advance and that no minimum interval between reservation and usage is necessary.

At stream establishment time only one QoS which is valid for the whole interval can be given; however, during renegotiations the interval boundaries can be changed and parts of the reserved interval can get a different QoS. Additionally, time intervals directly following the current reservation can be reserved to extend the overall stream duration. This both allows for the adaptation to changing requirements and considers the dynamic character of future applications. Alternatively, it could have been implemented that at set-up time several intervals with varying QoS might be specified, yet, we felt that such a feature would be seldomly used and hence did not justify the induced overhead.

We support only two partitions, one for immediate reservations, the other for advance reservations. The boundary between them can be changed. The watermarks of the partitions are fixed inside the reservation management.

The state of the RMS about reservations, etc., must be saved in permanent storage to provide means for failure recovery and for shutdowns before maintenance operations. In our prototype, its state is only saved either as part of its shutdown or on request via an API call provided for that purpose. This must be enhanced for a production system so that the state is either saved periodically or in case of state changes.

If a capacity reduction occurs so that not all established reservations can be served anymore then the reservations which are farest in the future are marked as `can't be met` first. If later the resources are again sufficient to serve reservations which have been marked as `can't be met` then the reservations which are closed to their start will be reactivated first. This straightforward approach tries to keep the currently active streams serviceable and to move the resource allocation for not active streams so far that they might become serviceable due to release of resources by currently active streams. Priorities might be added as further decision criterion in the future.

4.8 Module Structure

The ReRA RMS prototype is implemented as a server (Figure 12). The server stub calls functions of the notification component which communicates with the reservation management. The latter uses the services of the resource management component and loads the information about streams and their priorities into the scheduler when a new time slice begins.

The ReRA library on the client side offers APIs to set-up, to allocate, or to free reservations, to renegotiate QoS or reservation duration, to retrieve state and also statistical information, and to register for and perform functions of the notification service.

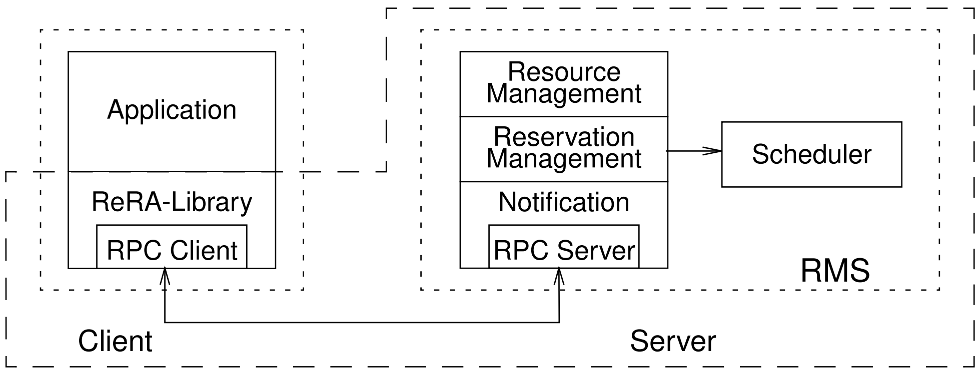

Figure 12: Module Structure.

4.9 Prototype

Figure 13 shows a screen shot of the user agent as user interface to the management component. The vertical axis is divided into areas for announcements, for reservations, and for statistical information about the resource usage. Additionally, menus for the announcements and reservations (generation, query, control, ...) are present on the screen.

Private and public announcement are kept separately and are not displayed at the same time, yet, the user can toggle between them. Own, new announcements are generated by filling in a form.

The list of reservations contains two immediate and one advance reservation. The statistics area illustrates the corresponding resource utilization. The immediate reservations occurred within a short time interval leading to the first two steps on the left side. The time of the release of the resources is unknown, hence, the utilization does not drop below that value. The advance reservation increases the resource utilization after several time ticks. Its duration is known in advance (leading to the displayed 'drop' after some time).

5 Related Work

In the last few years, several publications detailed the need for ReRA and discussed some ideas on appropriate mechanisms: the necessity for ReRA has become evident. However, to our knowledge only little work has been performed on that subject yet.

Roberts and Liao present in [15] a mathematical model for a reservation process and calculate the capacity requirements of video-conference networks in order to avoid bottlenecks and severe resource contention. This work is directed towards the (at the time of that article) upcoming ISDN networks. Hence, its applicability to computer-based multimedia systems is somewhat limited due to the different application and usage characteristics and the restricted system environment.

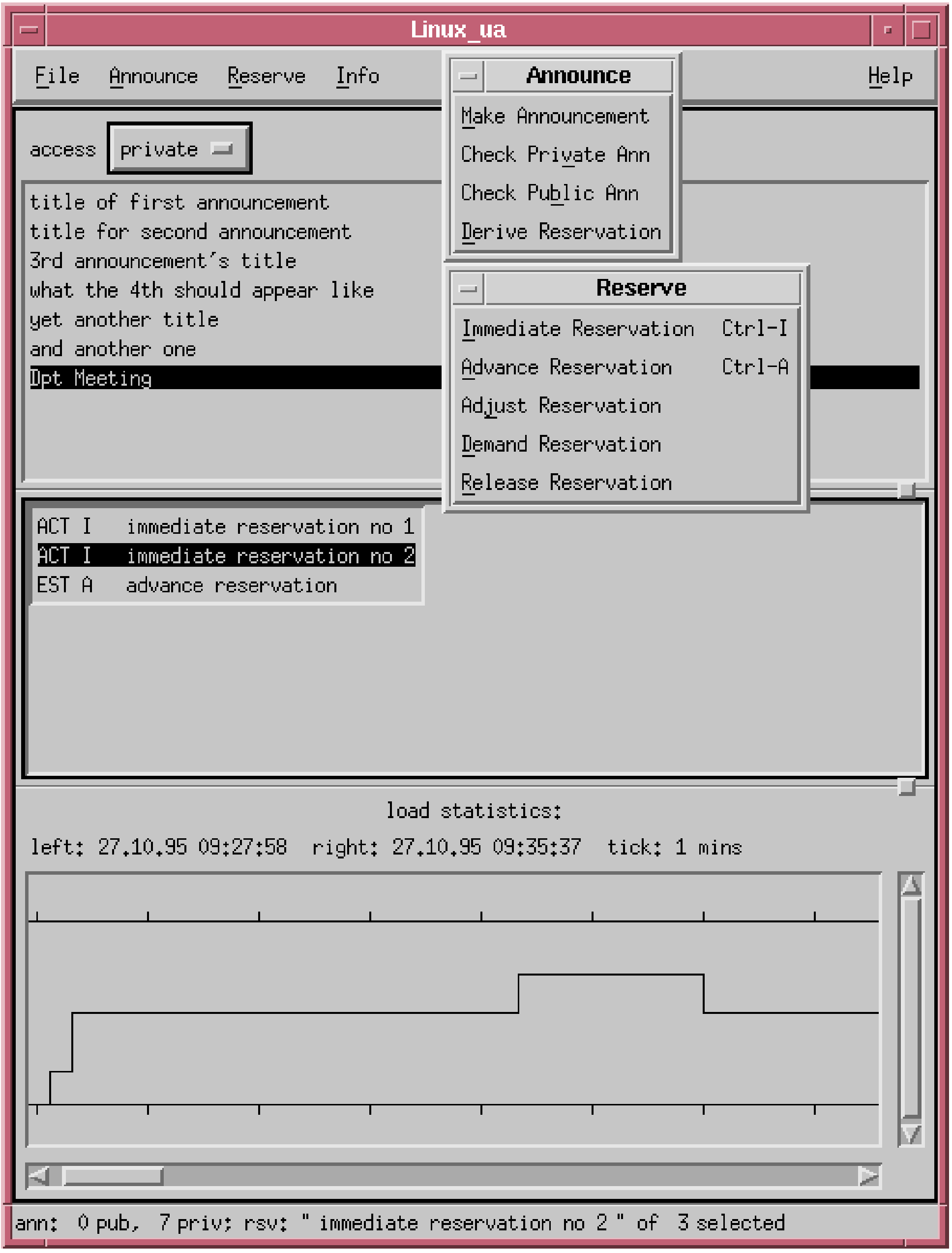

Figure 13: Initial Version of the User Agent.

Ferrari, Ramaekers and Ventre are among the pioneers who notice in [9] that ReRA is a useful concept. They describe the parameters for the start and duration of a reservation. They also discuss that the resource management system (only considering the network but not any other resources such as crucial resources involved in the processing within a video-on-demand server) must verify that a client does not exceed the time interval granted to him. Neither further requirements of ReRA nor design aspects are described in more detail.

Campbell, Coulson and Hutchinson describe in [2] their Quality of Service Architecture and specify start and end time parameters for 'forward reservations', however, they also state that these parameters have been omitted and remain for further study.

Chang presents in [3] a scheduling service based on the telephone system as basic model: Conferences are set up via a special service offered by a (phone system) service provider. The work describes a programming interface which offers functions for the creation and deletion of conferences (`conference_schedule()` resp. `conference_cancel()`) as well as for announcement of new conferences (`conference_announcement()`) and for searching (`conference_search()`) of created conferences. It is proposed to use X.500 for the provision of this function. Chang identifies (1) that the scheduler components of a resource management system can be kept unchanged and (2) that the reservation database and the schedulability test algorithms must be changed in order to take the time parameters into account. However, it is not discussed how this can be done. In Chang's work, all reservation requests must contain information about begin and duration of the reservation, yet it is not described what happens if a conference lasts longer than specified.

Reinhardt gives in [13] a straight-forward extension of the resource reservation protocol ST-II [17] to exchange (within the flow specification) the necessary information about start and end time of a reservation in advance and describes some problems to be solved within ReRA. In [14], he discusses application classes which can make use of ReRA and evaluates the resource reservation protocols ST-II and RSVP with respect to their suitability to reserve resources in advance. However, no general model is presented.

The NOSSDAV 1995 workshop contained three papers about ReRA [7, 5, 20]. Ferrari, Gupta and Ventre from the Tenet group at the University of Berkeley describe in [7] a scheme for advance reservations of real-time connections without a general architecture. They present methods for connection establishment where the usage duration must only be stated by advance reservations; 'immediate channels' are established for an undetermined amount of time. To avoid conflicts with advance reservations (resources already reserved for the latter cannot be used for reservations of unknown duration since then no guarantee about the availability of these resources can be given) the resources are partitioned into separate areas for immediate and for advance reservations. The boundary between the areas can be varied within certain constraints (watermarks). The authors developed also a mechanism to manage effectively the table of all set advance reservations by dividing the time into intervals which describe regions of constant resource availability.

Degermark, Köhler, Pink and Schelén show in [5] an extension of the admission control algorithm for predicted service suggested by Jamin, Clark, Shenker and Zhang [4, 10]. Furthermore, they discuss briefly how such a service can be implemented using RSVP [22] by periodically repeating RSVPs PATH and RESERVATION messages (already in advance to set up and keep the reservation state) and adding time parameters to the flow specification. A general model or architecture is not presented in their work.

Wolf, Delgrossi, Steinmetz, Schaller, and Wittig present in [20] a model for resource reservation in advance. In that paper, the issues to be resolved in resource reservation in advance systems have been discussed and it has been shown how the resource reservation in advance scheme can be embedded in a general architecture. This was the initial work which has been enhanced since then to serve as a basis for this publication. Many of these initial ideas have been refined during the design and implementation of the architecture with its prototype presented in Section 4.

6 Conclusions

While current resource management systems provide mechanisms which offer reliability with respect to QoS, this is not sufficient since many well established application scenarios, e.g., from the cooperative work area, are not well supported. Only with the advent of ReRA mechanisms several cooperative work applications will be accepted in our daily life.

ReRA is more than a simple extension of current resource reservation systems. As part of the development of ReRA systems, several issues must be attacked: The integration of current reservation schemes with ReRA requires resource partitioning methods. Applications must be offered a variety of mechanisms to prolong and adapt reservations. Failure handling raises difficult questions and must be carefully integrated into the system architecture. The provision of reservation mechanisms remains to be only one issue of a complete ReRA system. Agents to interact with the user, for reservation request generation as well as for the presentation and handling of incoming invitations, are necessary.

We presented an architecture which addresses such issues and offers suitable ReRA functionality. Our implementation shows that it is possible to provide ReRA capabilities to time constrained multimedia applications.

Acknowledgments

The practical work performed by Gerhard Sittig on ReRA provides the base of the architecture and implementation. His contributions together with fruitful discussions with Winfried Kalfa are gratefully acknowledged. Further thanks are due to our former colleagues Luca Delgrossi, Sibylle Schaller and Hartmut Wittig who participated actively in the early discussions and a preliminary report about ReRA.

References

[1] A. Banerjea, D. Ferrari, B.A. Mark, M. Moran: "The Tenet Real-Time Protocol Suite: Design, Implementation, and Experiences", Technical Report TR-94-059, International Computer Science Institute, Berkeley, CA, USA, November 1994.

[2] A. Campbell, G. Coulson, D. Hutchinson: "A Quality of Service Architecture", ACM Computer Communication Review, Vol. 24, No. 2, April 1994, pp.6-27.

[3] Y.-H. Chang: "Network Support for a Multimedia Conference Scheduling Service", Proceedings of SPIE, Vol. 2188, 1994, pp. 109-119.

[4] D. Clark, S. Shenker, L. Zhang: "Supporting Real-Time Applications in an Integrated Packet Services Network: Architecture and Mechanisms", SIGCOMM 1992.

[5] M. Degermark, T. Köhler, S. Pink, O. Schelén: "Advance Reservation for Predicted Service", Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH, USA, April 19-21, 1995.

[6] L. Delgrossi, L. Berger (Ed.): "Internet STream Protocol Version 2 (ST2) – Protocol Specification – Version ST2+", Internet RFC 1819, August 1995.

[7] D. Ferrari, A. Gupta, G. Ventre: "Distributed Advance Reservation of Real-Time Connections", Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH, USA, April 19-21, 1995.

[8] D. Ferrari, A. Banerjea, H. Zhang: "Network Support for Multimedia: A Discussion of the Tenet Approach", Technical Report TR-92-072, International Computer Science Institute, Berkeley, CA, USA, October 1992.

[9] D. Ferrari, J. Ramaekers, G. Ventre: "Client–Network Interactions in Quality of Service Communication Environments", Proceedings of the Fourth IFIP Conference on High Performance Networking, University Liege, Belgium, December 1992, pp. E1-1 - E1-14.

[10] S. Jamin, D. Clark, S. Shenker, L. Zhang: "Admission Control Algorithm for Predictive Real-Time Service", Third International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, CA, USA, November 1992.

[11] C.L. Liu, J.W.Layland: "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", Journal of the ACM, Vol. 20, No. 1, January 1973, pp. 47-61.

[12] K. Nahrstedt, R. Steinmetz: "Resource Management in Networked Multimedia Systems", IEEE Computer, Vol. 28, No. 4, April 1995.

[13] W. Reinhardt: "Advance Reservation of Network Resources for Multimedia Applications", Proceedings of the Second International Workshop on Advanced Teleservices and High-Speed Communication Architectures, Heidelberg, Germany, September 26-28, 1994.

[14] W. Reinhardt, "Advance Resource Reservation and its Impact on Reservation Protocols", Proceedings of Broadband Island'95, Dublin, Irland, September 1995.

[15] J.W. Roberts, K. Liao: " Traffic Models for Telecommunication Services with Advance Capacity Reservation", Computer Networks and ISDN Systems, No. 10, 1985, pp. 221-229.

[16] R. Steinmetz: "Human Perception of Jitter and Media Synchronisation". IEEE Journal on Selected Areas in Communications, Jamuary 1996., Vol. 14, No. 1, pp. 61-72.

[17] C. Topolcic: "Experimental Internet Stream Protocol, Version 2 (ST-II)," Internet RFC 1190, October 1990.

[18] C. Vogt, R.G. Herrtwich, R. Nagarajan: "HeiRAT: The Heidelberg Resource Administration Technique - Design Philosophy and Goals," Kommunikation in Verteilten Systemen, Munich, Germany, March 3-5, 1993.

[19] C. Vogt, L.C. Wolf, R.G. Herrtwich, H. Wittig: "HeiRAT — Quality-of-Service Management for Distributed Multimedia Systems", to appear in ACM Multimedia Systems Journal – Special Issue on QoS Systems.

[20] L.C. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, H. Wittig: "Issues of Reserving Resources in Advance", Fifth International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH, USA, April 19-21, 1995.

[21] L.C. Wolf: "Resource Management for Distributed Distributed Multimedia Systems", Kluwer, 1996.

[22] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala: "RSVP: A New Resource ReSerVation Protocol," IEEE Network, September 1993, pp. 8-18.