A scheme for time-dependent resource reservation in QoS-enabled IP networks♦

Roberto Canonico, Simon Pietro Romano, Mauro Sellitto and Giorgio Ventre

Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II", Napoli, Italy

Email: {canonico, sprom, sellitto, ventre}@grid.unina.it

Abstract. A number of distributed applications require communication services with Quality of Service (QoS) guarantees. The QoS provisioning issue in the Internet has been addressed by the IETF with the definition of the Integrated Services (IntServ) and Differentiated Services (Diffserv) frameworks. Resource reservation mechanisms on which these models are based are totally time-unaware. Yet, we believe that, in some cases, associating a time interval to network resource reservations could be useful for both users and network providers. In this paper we present a distributed scheme for time-dependent reservations in QoS-enabled IP networks. We also show how the standard signalling protocol RSVP may support this new reservation style, with only a few minor modifications. Finally, we present a first prototype implementation of the major component of the proposed architecture and we provide some hints on future applicability scenarios of the advance reservation paradigm and its impact on related topics such as policing and charging techniques in QoS-enabled IP networks.

1 Introduction

Existing communications systems are rapidly converging into an ubiquitous information infrastructure that does not distinguish between computing and communications, but rather provides a set of distributed services to users [1]. In this scenario the ability of the network to provide the applications with end-to-end Quality of Service (QoS) becomes a crucial issue. The best candidate infrastructure to support these new distributed services is the Internet, due to the enormous number of hosts already connected world-wide. Unfortunately, the communication service provided by current Internet does not offer QoS guarantees to applications. Having recognized the necessity of extending the basic best-effort service, the Internet Engineering Task Force has defined two architectures: *Integrated Services* [2] and *Differentiated Services* [3]. In particular, the former architecture enables applications to demand per-flow end-to-end guarantees from the network. To provide applications with the

♦ This work has been partially supported by the Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST), Italy, in the framework of project MOSAICO

required QoS, all network routers along the path between sender and receiver must be kept informed about the characteristics of the flows, so to reserve the necessary resources. A resource reservation protocol is thus needed to distribute this information over the network. For this purpose, the IETF has defined an appropriate signalling protocol, RSVP (*Resource reSerVation Protocol*) [4], whose role is to install a per-flow state in network routers, so to maintain a per-flow resource reservation along the communication path. To keep the connectionless approach of the Internet Protocol, RSVP mechanisms have been designed to continuously refresh resource reservations (*soft state*).

A major limitation of the IETF model is that it is totally time-unaware, i.e. reservations take place immediately and last for an unspecified duration. Many authors have already discussed the usefulness of an Advance Reservation Service [5, 6, 7, 8, 9]. In [6], for instance, Reinhardt focuses on the importance of this new kind of service for modern video conferencing and Video on Demand applications, whereas in [7] Wolf and Steinmetz also indicate manufacturing control systems and remote surgery as possible scenarios. In our opinion, the importance of time-dependent reservations in a QoS-capable Internet is of extreme importance because it allows a more efficient resource management in the network and can have a strong impact on pricing policies.

In this paper we compare some of the proposals that have emerged so far to support advance reservations in QoS-capable networks, and we present a new approach for Advance Reservations management, which is totally compatible with the Integrated Services Architecture. We also illustrate an implementation framework for our scheme, which relies on only a few minor changes to the RSVP protocol.

The rest of the paper is organized as follows. In Section 2 we discuss issues arising when time-dependent resource reservations are introduced in a QoS network. In Section 3 we present a scheme for distributed advance reservation in IP networks. In Section 4 the inner structure of the main component of our architecture is illustrated. Section 5 is devoted to the presentation of a prototype implementation of this component. Finally, Section 6 provides conclusions and discussion of future work.

2 Issues related to time-dependent reservations

A number of architectures have been already proposed to introduce Advance Reservation Services in a QoS network. A thorough description of them is presented in [7]. Some common architectural elements can be found in most of the proposed models. Both [8] and [9] propose a model relying upon a centralized approach. More precisely, they may be defined *server-based* as well as *routing-domain-based* since, for each Internet Autonomous System, they designate a host, named *Advance Reservation Server (ARS)* in [9] and *Advance Reservation Agent* in [8], whose purpose is to receive and manage all advance reservation requests. These works focus on the problems associated to the implementation of a *Reservation Management System.*

When adopting a server-based solution, an application may ask the ARS to set up an advance reservation and, after receiving a confirmation message, may terminate

without any need to be active for the entire duration of the *bookahead* period. In this scenario, all state information relating to the advance reservation may be stored into a stable memory (*hard-state*), thus eliminating all the problems associated with the existence of long-lived soft-state inside the routers.

A centralized solution has the advantage of relieving the routers from the burden of performing all the actions related to Admission Control. In fact, the introduction of an appropriate server relegates the routers to their normal forwarding and scheduling tasks. Nevertheless, the resulting architecture also shows the typical disadvantages of centralized systems: the creation of a performance and reliability bottleneck, poor scalability, and the need to keep in the centralized reservation server a consistent and up-to-date view of the present and future resource allocations throughout the network [5]. Furthermore, most of the signalling traffic associated to reservations would be concentrated on server nodes, causing them to become possible bottlenecks inside the network. A distributed solution, on the other side, shows a high degree of robustness with respect to crashes and uniformly allots control traffic across network nodes. The major drawback of such a solution remains in the augmented complexity of the routers.

All the proposed models for Advance Reservation Services rely upon a signalling protocol. For this purpose, an extension to RSVP seems both natural and simple to achieve. One could think of defining a new object used to specify *Time* as an additional QoS parameter. Such an object would be transferred across the network into RESV messages [10]. However, a number of issues arise, due to the specific mechanisms upon which the protocol relies. In particular, the main limitations deal with the receiver-oriented nature of RSVP and the soft-state paradigm that it adopts [6].

In the following we describe a distributed approach for management of time-dependent resource reservations which aims at introducing only a few modifications to the IETF Integrated Services Architecture. Our approach relies upon an augmented version of RSVP.

3 A distributed Advance Reservation scheme

Scope of this paper is to present a distributed solution for the implementation of a *Reservation Management System* [7], where the role played by RSVP is of crucial importance. It is our firm belief, in fact, that the advantages of such a solution largely overwhelm the minor drawbacks associated to router complexity.

In the architecture we propose, network elements (routers and hosts) assume the structure showed in Figure 1. The basic idea is to provide a *Reservation Manager* that takes up the responsibility of processing advance reservation requests during the negotiation phase, while relying on the well-known mechanisms of RSVP during the usage phase. Its main purpose is to perform admission control, *i.e.* to check whether enough bandwidth is available for the whole duration of the reservation interval. State associated to each accepted reservation is stored in a *Reservation-DB*, which also holds the information concerning bandwidth availability for each time slot. In this model, we cope with both immediate and advance reservations in a fashion that

proves to be as less intrusive as possible with respect to existing RSVP-based QoS frameworks. Immediate reservations, in fact, are handled according to the standard *IntServ* model, with the only difference residing in the admission control module, which is now in charge of taking into account the resources allocated both to immediate and advance reservations. However, the negotiation phase for an advance reservation is under the responsibility of *Reservation Managers*, which make use of both standard and *ad hoc* defined signalling messages.

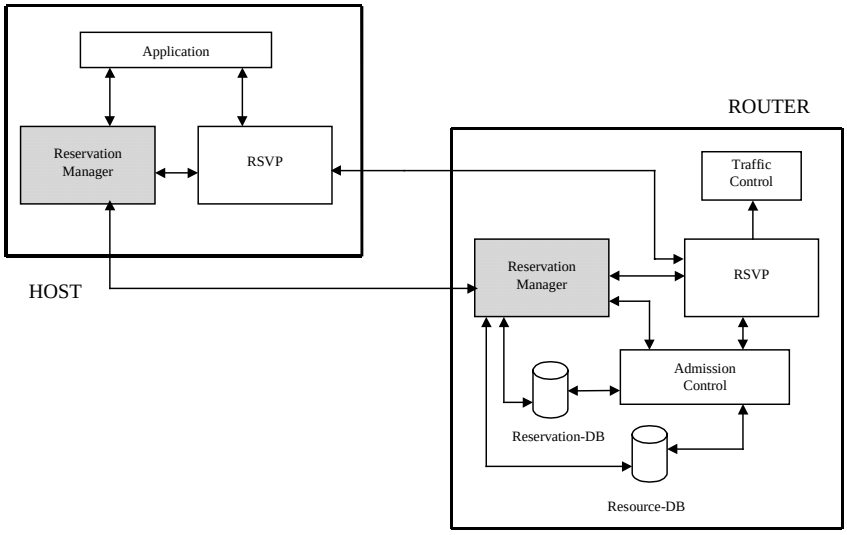

Fig. 1. Hosts and routers structure

Figure 2 depicts the steps associated with the negotiation of an advance reservation.

1. The receiving application sends the local *Reservation Manager* its reservation request.
2. To characterize the path between itself and the sender, the *Reservation Manager* of the receiving host executes a *rapi_session()* call [11] to the local RSVP daemon. In order to tell its counterpart on the sending side to start sending path characterization messages, a new message is defined, called PATH_REQUEST, which flows on the upstream direction from receiver to sender.
3. To appropriately trace the route between sender and receiver, the *Reservation Manager* on the sender side makes use of a PATH_TRACE message containing an object similar to the standard RSVP_HOP object. As it travels downstream, the PATH_TRACE message is used to install path information in intermediate routers' *Reservation Managers*, needed to forward advance reservation messages in a further step.
4. Sender's *Reservation Manager*, based on the information contained into the PATH_REQUEST message it received, executes both a *rapi_session()* and a *rapi_sender()* call, thus triggering the standard RSVP PATH sending phase. PATH messages flowing in the downstream direction are intercepted also by the

Reservation Managers of intermediate nodes, which use them to build messages associated with advance reservations (ADV_RESV messages).

5. Receiver's *Reservation Manager* notifies the receiving application of the arrival of PATH messages from the sender, thus allowing the application to make all the computations required to determine the parameters related to the desired QoS. The application then executes an appropriate *advance_reserve()* call which triggers the creation and forwarding of an ADV_RESV message from the receiver's *Reservation Manager* to the first router upstream.
6. Upon reception of the ADV_RESV message, the router's *Reservation Manager* performs an admission test by calling an appropriate function (*Add_New_Adv_Resv()*); if this test is passed, the ADV_RESV message is further forwarded to the next upstream router, eventually reaching the sending host.
7. In case of positive response from all admission tests along the path, the receiver's *Reservation Manager* is notified with an ADV_RESV_CONF message, whose purpose is to confirm reservation acceptance; this in turn provides the application with a specific reservation identifier that will be used in the following phase associated with resource usage.

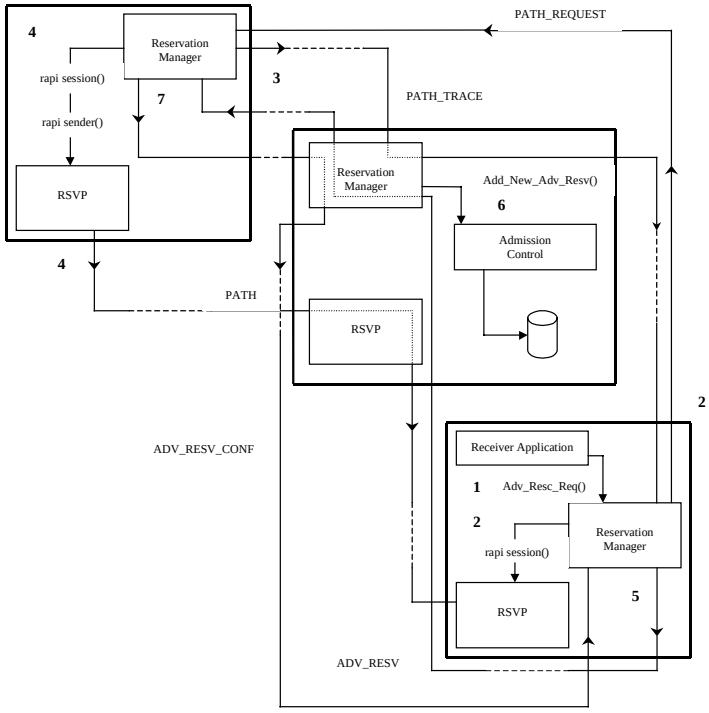

Fig. 2. The negotiation of an Advance Reservation

After the negotiation phase for an advance reservation, there is no need for the application that made the request to stay active even during the intermediate phase.

At the beginning of the usage phase, *Reservation Managers* automatically label the required resources as allotted to a pending reservation: data structures necessary to carry on classification and scheduling tasks are actually instantiated only when an explicit usage request from the involved application arrives. For this purpose, the receiving application uses a slightly modified version of the standard *rapi_reserve()* call, containing a new parameter related to the identification of an already accepted and registered advance reservation. As a consequence, usage requests for advance reservations are easily handled as immediate reservations for which admission control tests consist in just verifying the existence, in the *Reservation-DB*, of an advance reservation with the same identifier as the one provided by the *rapi_reserve()* call.

Our model is thus able to cope with both the major problems associated with the use of RSVP in an advance reservation framework, *i.e* soft state and receiver-oriented approach. The issue related to the possible absence of the sender is, in fact, accommodated by the introduction of the *Reservation Manager* entity, whereas all problems concerning the soft state are eliminated thanks to the Reservation and Resource databases, which guarantee that, for the entire duration of the intermediate phase, no soft state has to be managed by network nodes.

4 Reservation Manager mechanisms

The goal of the *Reservation Manager* is to cope with the absence, in RSVP, of a set of suitable mechanisms to handle reservations in advance. In the following we analyze in more detail some of the relevant characteristics of the overall framework.

4.1 Negotiation handshaking

The main function of *Reservation Managers* is to process advance reservation requests. To accomplish this task, they exchange a number of messages needed to decide whether or not a specified request must be accepted and register state information inside the routers involved in the reservation process. Messages exchanged during this phase are: PATH_REQUEST, PATH, PATH_TRACE and ADV_RESV, as showed in Figure 3.

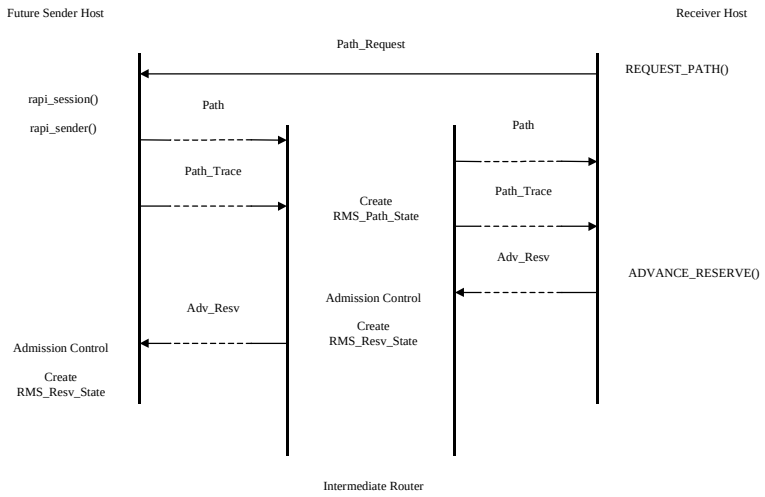

Fig. 3. Typical handshaking for the negotiation phase of an Advance Reservation

It is worth noting the introduction of a PATH_REQUEST message, to deal with the possible absence of the sender at the time when an advance reservation request is made from a potential receiving application. With this new message the receiving application may solicit the *Reservation Manager* at the sender side to start sending PATH messages, thus acting as if the actual sender were present. These PATH messages, however, are only needed to gather all the information related to the path from sender to receiver (contained, for example, in the ADSPEC object). The specific format of this and other new messages introduced is detailed in [12].

Faults, as well as re-routings, occurring during the intermediate phase between negotiation and usage should be taken into account by the *Reservation Managers* with the help of *Resource Managers;* nevertheless, these issues are beyond the scope of a trial implementation and we left it for further development of the overall framework.

4.2 Confirmation of reservation acceptance

Each node, upon reception of an ADV_RESV message**,** has to submit the request to the Admission Control module. If the test is passed successfully, a local reservation identifier is returned and a RMS_Resv_State object is built with the purpose of storing all the information related to the advance reservation. This object is inserted into the reservation database only when admission control has been successful on each and every node along the path. This requires the introduction of a new message, especially tailored to accomplish this task. This new message, called ADV_RESV_CONF, is showed in the lower part of Figure 4.

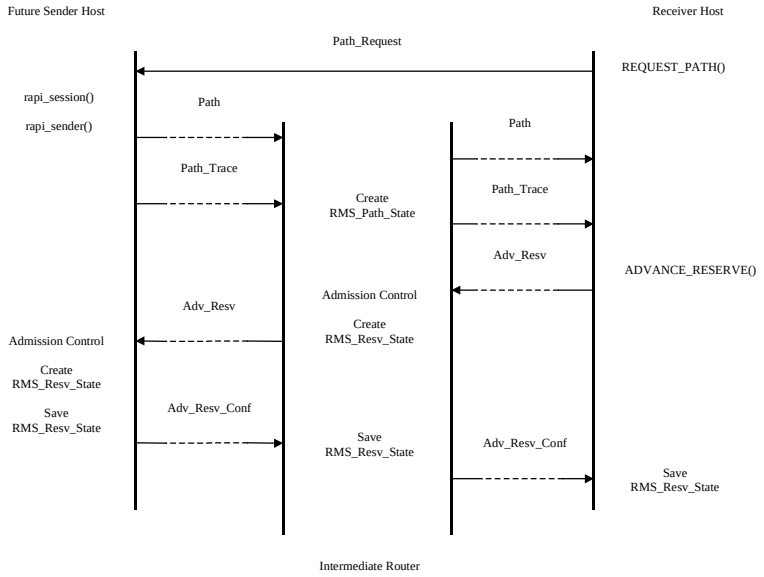

Fig. 4. Confirmation message for advance reservations

4.3 Flowspecs merging

Flowspecs merging in the case of advance reservations introduces a new dimension with respect to immediate reservations. No merging is possible if the time intervals of two different reservations do not intersect. Things change when at least a partial overlapping exists, as depicted in Figure 5.

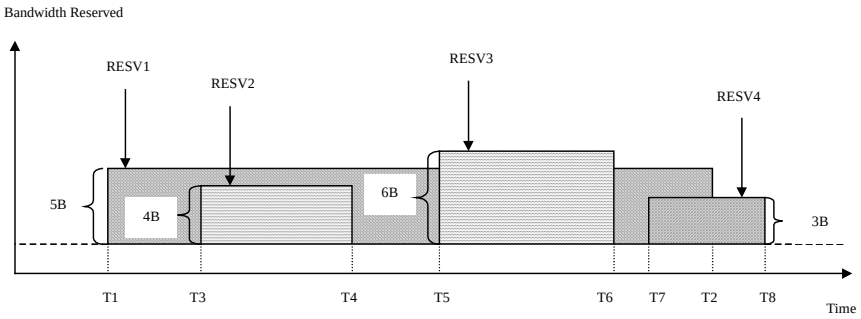

Fig. 5. Flowspec merging in the case of advance reservation

A simple analysis of Figure 5 shows that merging is allowed if and only if the new request arriving at a node is “fully included” into a reservation already in place, *i.e.* if its *flowspec* turns out to be “less than or equal to” [4] the already existing one and the reservation interval for the latter request is included into the time interval of the

former. The admission test for a new request, however, must always check whether the interested flow had already been assigned a portion of resources, even in a sub-interval of the one conveyed inside the request. For instance, when RESV3 arrives (Figure 5), the bandwidth considered for admission test must be B, instead of 6B, since the flow has already been assigned a bandwidth of 5B with RESV1.

4.4 Reservation cancellation

An advance reservation may be cancelled from the requesting application via the *delete_adv_resv*() function, which causes all the state information stored into the receiver's database to be deleted. To achieve the same goal on all the nodes along the path, the *Reservation Manager* at the receiving side sends upstream a DEL_ADV_RESV message that triggers the cancellation process on each intermediate element while crossing the network. The cancellation process becomes a little tricky in the presence of merged reservations. A thorough discussion of the issues involved in this case may be found in [12]. Furthermore, it should be considered the case of a lost DEL_ADV_RESV, resulting in persistance of state information associated to a deleted reservation into some of the intermediate nodes' databases. To solve this problem, an acknowledgement message, called DEL_ADV_RESV_ACK, has been introduced, with the purpose of confirming the cancellation of a specified reservation. The reservation cancellation process is showed in Figure 6.

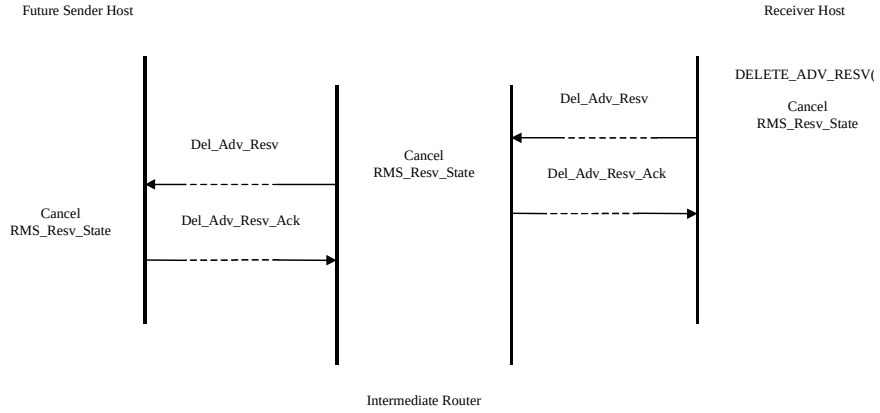

Fig. 6. Reservation cancellation

4.5 Error handling

An advance reservation request which fails the admission control test on a node, causes an error message, called ADV_RESV_ERR, to be generated and sent downstream towards the receiving host. Each intermediate node that gets this message deletes all state information concerning the request, whereas the receiving host also

uses an upcall routine to inform the receiving application about the error. The same technique is used to deal with other errors, such as those relating to the absence of a suitable RMS_Path State object for a session, as well as to the presence of an unknown or conflicting reservation style.

4.6 QoS Enforcement phase

QoS enforcement for advance reservation does not differ from the setup of an immediate reservation. As already mentioned, this new kind of reservations may be seen, during this phase, as usual immediate reservations for which the admission control test simply consists in a check into the *Reservation-DB* aiming at verifying the existence of a RMS_Resv State object matching the parameters contained in the usage request. This object includes information concerning the time interval for which the reservation is being requested and a handle for reservation authentication inside the node database. These new parameters must thus be included in the usage request too. This is accomplished by exploiting the RSVP standard interface by means of the *reserve*() function, slightly modified to the purpose. The execution of such primitive causes modified (*i.e.* including time interval and reservation handle) RESV messages to be generated at the receiving application's side**.** Upon reception of this modified RESV message, each node is able to realize whether the request is for an immediate or an advance reservation and may consistently choose the right admission control algorithm to be performed.

5 A prototype Reservation Manager

The *Reservation Manager* is responsible for the management of time-dependent reservations in the overall system. Its main role is admission control. State associated to each accepted reservation is stored in a *Reservation-DB*, which also holds the information concerning bandwidth availability for each time slot. The system is also responsible for monitoring reservation state changes, with special regard to transitions from established to pending states, and from pending to expired state [7]. The main components of the Reservation Manager are showed in Figure 7.

A time-dependent reservation may be required via the user interface provided by the `rms_main` module. Following a reservation request, the `admission_ctrl` module is activated to perform the admission test for that flow. To the purpose, it interacts with the *Reservation-DB*, both to retrieve information concerning bandwidth availability in the specified time interval and to store data structures associated to new reservations. The last module showed in the figure relates to timer management. It manages a sorted time slots list. An appropriate timer is set so to expire as soon as the time slot on top of the list starts, thus triggering the activation of a handler. This function is in charge of doing all the actions associated to the transitions from established to pending states (for reservations waiting to be activated) and from pending to expired (for reservations which were supposed to be activated during the previous slot).

A prototype of a Reservation Manager compliant with the model presented in Figure 7 has been implemented for the FreeBSD operating system. This module interacts with a traffic control module based on a WFQ packet scheduler [13].

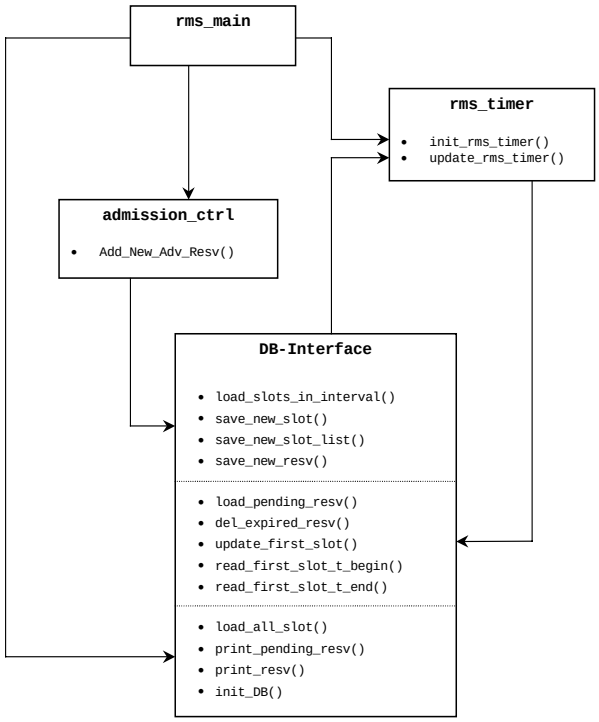

Fig. 7. Structure of the Reservation Manager

6 Conclusions and future work

In this paper we have presented a scheme for distributed advance reservation in QoS-enabled IP networks. Our original contribution consists in the design of a distributed solution both efficient and simple to implement. As for applicability is concerned, we went into the details of a solution especially suited to extend the IETF Integrated Services model. Since we are firmly convinced that, while taking into account user expectations and needs, it is compulsory to consider Quality of Service as an end-to-end issue, we assumed the IntServ model as a starting point. In our framework, we are now finished with the development of a prototype Reservation Manager and work is currently in full swing to integrate this new model inside current RSVP protocol implementations.

Work is in progress to design a model for Virtual Private Networks comprising both IntServ and Diffserv components and aiming at the definition of a general environment, exploiting advanced policing and charging mechanisms to provide QoS at different levels of granularity. More specifically, we envision the use of our signalling protocol both at a microflow level, as discussed in this paper, and at a higher hierarchical level, as a means for implementing the handshaking required between Bandwidth Brokers [14] for dynamic negotiation of Service Level Agreements. In this scenario the network is considered as an entity which is able to provide a service general enough to include both immediate and advance reservations while leveraging on charging to enforce differentiation according to specific service parameters.

References

1. D. Clark, J. Pasquale *et al.*. "Strategic Directions in Networks and Telecommunications". ACM Computing Surveys, No. 4, Dec. 1996, pp. 679-690.
2. R. Braden, D.Clark, and S. Shenker. "Integrated Services in the Internet Architecture: an Overview". *RFC 1633*, July 1994.
3. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. "An Architecture for Differentiated Services". *RFC 2475*, Dec. 1998.
4. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification". *RFC 2205*, September 1997.
5. D. Ferrari, A. Gupta, and G. Ventre. "Distributed Advance Reservation of Real-Time Connections." In Proc. of the Fifth Int. Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH, USA, April 1995.
6. W. Reinhardt. "Advance Resource Reservation and its Impact on Reservation Protocols". Proceedings of Broadband Island ’95, Dublin, Ireland, September 1995.
7. L. C. Wolf, R. Steinmetz. "Concepts of Resource Reservation in Advance". Special Issue of Multimedia Tools and Applications on “The State of The Art in Multimedia“, May 1997 (Vol. 4, No. 3).
8. O. Schelén, S. Pink. "An agent-based architecture for advance reservations". In IEEE 22nd Annual Conference on Computer Networks, Minneapolis, Minnesota, November 1997.
9. S. Berson, R. Lindell, R. Braden. "An Architecture for Advance Reservations in the Internet". USC Information Sciences Institute, July 16, 1998.
10. J. Wroklawsky. "The use of RSVP with IETF Integrated Services". *RFC 2210*, September 1997.
11. R. Braden and D. Hoffman. "RAPI -- An RSVP Application Programming Interface - Version 5". IETF Internet Draft draft-ietf-rsvp-rapi-01.txt, Aug. 1998.
12. M. Sellitto. "Uno schema per la riservazione time-dependent di risorse in reti IP a Qualità di Servizio". (In Italian). M.SC. Thesis, University of Naples, July 1999.
13. R.D'Albenzio, S.P. Romano and G. Ventre. "An Engineering Approach to QoS Provisioning over the Internet". Lecture Notes in Computer Science no. 1629, Springer, May 1999, pp. 229-245.
14. K. Nichols, V. Jacobson, and L. Zhang. "A Two-bit Differentiated Services Architecture for the Internet". Internet draft, `draft-nichols-diff-svc-arch-00.txt`, Nov. 1997.