Multicast Flow Aggregation in IP Tunnels with Resource Reservation *

Paulo Cesar S. Vidal (vidal@de9.ime.eb.br) and Otto Carlos M. B. Duarte (otto@gta.ufrj.br)

Abstract— **The Internet Integrated Services Architecture has a fundamental scaling problem due to the storage and processing of states for every flow at all routers and end-systems supporting a flow. Reducing resource reservation state overload of RSVP Protocol is an important issue in order to scale the Integrated Services. The technique of IP-in-IP tunneling can be used to create an flow aggregation region. Thus, flows of multiple sources that reaches the entry router of the aggregation region follow to the exit router of the aggregation region at the IP-in-IP tunnel in an aggregated way. Therefore, all interior routers in the aggregation region process an aggregate flow, instead of processing individual flows. Based on this technique, this paper proposes a state aggregation mechanism of the RSVP protocol for multicast flows in transit using IP-in-IP tunneling. The simulation results show the mechanism performance in process of reservation set-up when losses of the tunnel RSVP messages occur.**

I. Introduction

Several real-time multimedia applications have been developed in the Internet, where best-effort delivery model of Internet Protocol (IP) is inadequate. So, there is a need to provide multimedia applications with service classes of better Quality of Service (QoS) related to bandwidth, packet queuing delay, and packet loss. In response to the growing demand of new applications, the Internet Engineering Task Force (IETF) defined an Integrated Services architecture in the Internet [2], where classes of services are specified. Supported by routers, this architecture can provide QoS guarantees to each data flow. The QoS level provided by these classes is programmable per flow, in accordance with applications requirements. These requirements can be passed to routers using Resource Reservation Protocol (RSVP) [3]. Two service classes [4] have been specified to be used with RSVP protocol: the guaranteed service and the controlled-load service. Although RSVP protocol was originally designed for resource reservation, some proposals use RSVP protocol to carry other types of control information in the network, as security QoS [5] and label-switched paths in MPLS (Multi-Protocol Label Switching) [6]. Another example is the use of RSVP protocol within the Differentiated Services [7, 8].

A flow is identified by its unicast or multicast address and its state is stored in the routers between the source and the destination. A greater usage of integrated services means a greater amount of stored states in the routers and a large processing is needed to manage them, thus overloading the routers. Hence, the technique of flow aggregation has been used to reach the necessary scalability to provide integrated services, reducing the amount of stored states to unicast or multicast flows.

*This work was supported by IME, UFRJ, FUJB, PROTEM-CC, CNPq and CAPES.

Researches have been carried out to decrease of number of states and several aggregation mechanisms have been proposed [1, 9, 10, 11, 12].

The work of Terzis et al. [1] implements the RSVP protocol over IP-in-IP tunnels where several end-to-end unicast sessions are mapped into a tunnel session between two tunnel end points. The state reduction is achieved because the interior routers do not process end-to-end RSVP messages. Interior routers are placed between two tunnel end points, and handle only tunnel messages that carry aggregated data of the end-to-end flows. In this way, the interior routers store only the state of the tunnel sessions.

In [9] it is proposed an architecture for resource reservation, where applications perform reservation requests through an agent. For every routing domain in the network, there is an agent responsible for the admission control of reservations. The agent aggregates several requests from different source addresses in a group that has the same destination address (same domain). Thus, the agent stores one state composed of a destination network address and the sum of the bandwidth of all receivers.

Jens et al. [10] present a mathematical analysis of the use of aggregation for identical flows of guaranteed service, that is, the flows must have the same traffic characteristic and follow the same route up to their destination. The analysis shows that an amount of bandwidth needed to reserve a flow group is smaller than the addition of individual reservations.

Davie et al. [11] describes a mechanism that uses a single RSVP reservation to aggregate other RSVP reservations across a transit routing region, in a manner conceptually similar to the use of virtual paths in a ATM network. It proposes a way to dynamically create the aggregate reservation, classify the traffic for which the reservation applies and to determine how much bandwidth is needed to achieve the requirement.

Fisher et al. [12] presents a design of the Beagle signalling protocol to support of the temporal resource sharing. The protocol provides the sharing for the same set of resources among related flows over time leading to significant resource gains and a decrease in the number of reservation states.

This article proposes a state aggregation mechanism of RSVP protocol that identifies multicast flows in transit in a determined aggregation region and groups its states among boundary routers, using IP-in-IP tunneling. The aggregation region can be, for example, a Domain, an Autonomous System (AS), an Internet Services Provider (ISP), or a Virtual Private Network (VPN). The mechanism is an extension of the model of Terzis et al. [1], which adding functions to aggregate multicast flows. The multicast aggregation mechanism can present a very long

latency at reservation establishment when losses of the tunnel RSVP messages occur. Therefore, we propose also the use of staged refresh timers [13] to decrease the latency. The simulation results show the performance gain in reservation set-up delay of the aggregation mechanism when the staged refresh timers is used.

The article is organized in the following way: Section 2 shows RSVP protocol and Section 3 describes the model of RSVP tunnels for unicast flows. Section 4 describes the extended mechanism to aggregate multicast flows. Section 5 presents the results of aggregation mechanism simulation and Section 6 shows final considerations and future works.

II. RSVP Protocol

The RSVP protocol [14] can be used by an end station to request resources for an application data flow through guaranteed service class or controlled-load service class. The first class provides a bandwidth guaranteed level and an end-to-end delay limit. The controlled-load service reserves bandwidth, but does not provide delay guarantees.

The RSVP protocol can be applied to both unicast and multicast flows, adapting dynamically to the group membership changes as well as to the route changes. It is receiver-oriented, that is, the receiver of a data flow initiates and maintains the resource reservation used for that flow. The RSVP protocol identifies a flow session by combining the destination address, the type of transport protocol and the destination port number.

The sequence used by RSVP protocol to establish and to maintain a reservation for a flow is realized through the following steps:

1. The source sends *Path* messages towards the receiver(s). These messages construct a path, maintain a "path state" in each router between the sender and the destination and advertise the characteristics of the flow. This path is constructed through of the PHOP object (*previous-hop*) carried out by *Path* messages. The RSVP messages are composed by fields called RSVP objects.

2. The receiver sends *Resv* messages conducting the reservation request towards the source. These messages must follows exactly the reverse of the paths the data flow will use. These messages use the information of the PHOP field storaged in "path state". They create and maintain reservation states in each router along the path.

3. Every node periodically sends *Path* and *Resv* messages (refresh messages) to maintain the states along the reserved path. Associated with each reserve state entry is a cleanup timer. The state is deleted if no refresh messages arrive before the expiration of a *cleanup timeout* interval (default value is 90s). This mechanism is called soft-state mechanism.

The soft-state mechanism provides good support for dynamic membership changes and automatic adaptation to routing changes. However, the soft-state mechanism does not have reliability in message delivery and it can lead to several problems in occurrence of message losses. If the first *Resv* message from the user is lost due to network link noise or congestion, the message will not be retransmitted until the next refresh cycle arrives. By default, a refresh interval R is 30 seconds. This can result in a very long latency at reservation establishment.

In Internet Multicast Backbone (MBone), the average packet loss rate is around 1-2%, and can occasionally be as 20% on congested links [15]. In this case, the existing delivery method of RSVP messages may not be acceptable for certain applications.

Pan and Schulzrinne [13] proposed a feedback mechanism called *staged refresh timers* to support reliable delivery of the RSVP messages and also to reduce the time needed in the resource reservation establishment of flows.

After sending the initial *Path* message, an RSVP node will wait for a lapse of time called initial retransmission interval (default value is 3s). If by that time a *PathAck* message has not been received, the node retransmits the *Path* message. The retransmission interval is controled by a staged refresh timer and it increases exponentially until it reaches a threshold (refresh interval R). The value is multiplied by a factor $(1 + \Delta)$ at each retransmission of a *Path* message. The default value used for Δ is 0.3. The retransmission will continue until a *PathAck* message arrives. Therefore, the period of the flow establishment is reduced to the initial retransmission interval plus the round-trip-time (RTT) of messages when an only *Path* message is lost. The process of *Resv* message retransmission occurs of the same way that in *Path* messages.

III. Aggregation in IP-in-IP Tunnels

The technique of IP-in-IP tunneling [16] can be used to create an flow aggregation region. This way, the flows of multiple sources that reaches the entry router of the aggregation region follow to the exit router of the aggregation region inside of the IP-in-IP tunnel in an aggregated way. Therefore, all interior routers in the aggregation region process an aggregated flow, instead of processing individual flows. Based on this technique, the work of Terzis et al. [1] implements an aggregation mechanism of RSVP protocol states over tunnels for unicast flows.

The mechanism of RSVP tunnels is shown in Figure 1, where the sender and the receiver of an end-to-end RSVP session are connected by the tunnel. The R_{entry} router is a tunnel entry router which encapsulates and sends the datagrams to the R_{exit} through the interior routers. The R_{exit} router is the end point of the tunnel that receives, decapsulates and sends the datagrams based upon the original inner IP header. In this mechanism there are two different sessions: a tunnel RSVP session and an end-to-end RSVP session. The end-to-end RSVP session is a individual session with resource reservation between sender and receiver. The tunnel RSVP session placed between the R_{entry} and the R_{exit} points provides resource reservations for a group of individual RSVP sessions set up in the tunnel.

The *Path* and *Resv* messages of an end-to-end session that reach the tunnel ends are mapped into the tunnel session associated with an aggregated traffic class. They are encapsulated and decapsulated in the same way as packets IP. The outer header specifying the tunnel entry point as source and the tunnel exit point as destination is added. Figure 1 illustrates the mapping of an end-to-end RSVP session, but there can be other end-to-end sessions

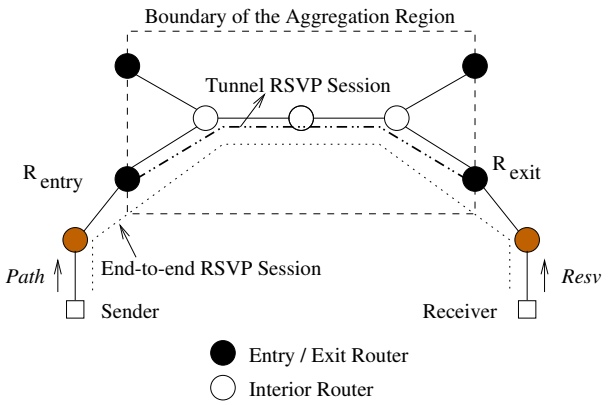

Fig. 1. Model of unicast RSVP tunnels

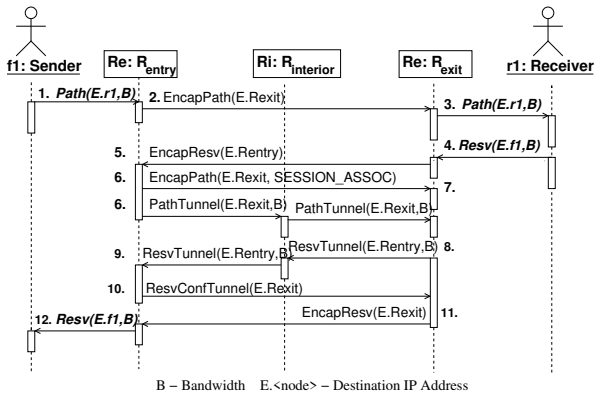

Fig. 2. Message Sequence Diagram of unicast RSVP Tunnel

associated with the tunnel RSVP session.

There is two possibilities to associate end-to-end sessions with corresponding tunnel sessions: to map all end-to-end sessions of same traffic class to a single tunnel session, or to allow multiple end-to-end sessions for each traffic class to be associated with a tunnel session. The association between end-to-end sessions and a tunnel session is made through the object SESSION_ASSOC inserted in the end-to-end *Path* message in the R_{entry} router. This object associates an end-to-end session with a tunnel session. The SESSION_ASSOC object has information about the end-to-end session (destination address, transport protocol and destination port number) and information about the tunnel session (address and port of the tunnel entry router). If there is only the simplest mapping of all the end-to-end sessions of the same traffic class to a single tunnel session, we eliminate the need for the SESSION_ASSOC object.

The interior routers do not recognize end-to-end RSVP messages, they handle only the RSVP messages generated by the end points of the tunnel. Therefore, there is a reduction in the number of states in the interior routers.

The sequence of the process of reservation set up in the tunnel is represented in Message Sequence Diagram [17] (Figure 2) and it is detailed here:

1. The R_{entry} router receives an individual *Path* message and creates the path state of the end-to-end RSVP session.

2. The R_{entry} router sends the encapsulated end-to-end *Path* message over the tunnel to the R_{exit} router.

3. The R_{exit} router receives the message, decapsulates it and creates the path state of the end-to-end RSVP session and forwards the original message to the destination.

4. When an end-to-end *Resv* message associated to an end-to-end RSVP session reaches the R_{exit} router, the router creates the reservation state, encapsulates and sends the message to the entry router.

5. The entry router receives the encapsulated *Resv* message, decapsulates it and creates the reservation state mapping the end-to-end RSVP session to a tunnel RSVP session.

6. The R_{entry} router sends the end-to-end *Path* message encapsulated with the SESSION_ASSOC object, associating the end-to-end RSVP session with the tunnel RSVP session to exit router. The entry router also sends the tunnel PATH message associated with the tunnel RSVP session to interior routers.

7. The R_{exit} router receives the *Path* message with the SESSION_ASSOC object, registers the association, removes the object and forwards the message to the destination.

8. The exit router receives the tunnel PATH message and sends a tunnel RESV message with the RESV_ CONFIRM object to the interior of the aggregation region, requesting resource reservation. The RESV_ CONFIRM object carries the exit router IP address that requested a confirmation.

9. If some interior router does not accept the reservation request, then it sends a tunnel RESV_ERROR message to the exit router. Thus, when the tunnel RESV message reaches the R_{entry} router, it means that the request was accepted by the interior routers.

10. After receiving the tunnel RESV message, the entry router sends a tunnel RESV_CONFIRM message to the R_{exit} router confirming the reservation in the tunnel.

11. The R_{exit} router receives the message and sends the encapsulated end-to-end *Resv* message to the R_{entry}.

12. The R_{entry} router receives the message, decapsulates it and sends the original end-to-end *Resv* message to the sender.

After the reservation set up, the exit router receives periodically *Resv* messages. However, an end-to-end *Resv* message can cause a change in individual reservation state parameters and in the associated tunnel session. If the change requires to increase the reservation, then R_{exit} router sends a tunnel RESV message including the RESV_CONFIRM object. If a RESV_CONFIRM message is received, the original *Resv* message is encapsulated and sent through the tunnel. If the tunnel reservation update fails, the R_{exit} router must send an end-to-end *Resv_Error* message to the originator of the end-to-end Resv message. If the change decreases the reservation, then R_{exit} router sends a tunnel RESV message without the RESV_CONFIRM object to update the reservation state in interior and entry routers.

When the entry router receives data packets, it verifies if there is resource reservation to data flow associated with the end-to-end RSVP session included in the packet. If there is no reservation for the packet, the packet is encapsulated and sent to the exit router. If there is reservation

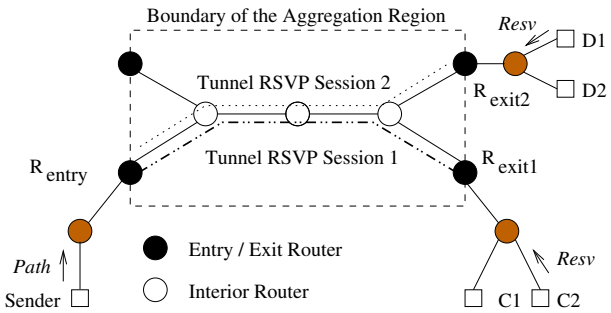

Fig. 3. Mechanism of multicast RSVP tunnels

to the flow, the data packet is encapsulated in IP and in User Datagram Protocol (UDP), because the individual sessions are differentiated by the source UDP port. Other components of the tunnel session, as the source IP address, the destination IP address and the destination UDP port are the same for all individual sessions.

The source UDP port is chosen by the entry router when it establishes the initial path state for a new tunnel session. The source UDP port associated with the new session is then conveyed to the R_{exit} router by the SESSION_ASSOC object.

IV. Multicast Aggregation Mechanism

The multicast flow aggregation mechanism proposed in this paper is applied to controlled-load service and aggregates the traffic in transit in the aggregation region by using the tunneling technique.

The mechanism is an extension of the model proposed by Terzis et al. [1] and the main changes introduced are:

- Diffusion of *Path* messages to several exit routers in the mapping of the end-to-end RSVP multicast session in tunnel RSVP sessions.
- Use of the staged refresh timers in the tunnel RSVP message retransmission.
- Marking the reserved data packets through the *Type of Service* (TOS) field or *Differentiated Services* (DS) field.
- Use of the destination multicast address in the IP reserved data packet outer header to decrease the bandwidth in aggregation region.

The association of the multicast RSVP session with several tunnel RSVP sessions is illustrated in Figure 3. Figure 3 shows a multicast tree formed by a source and four receivers (C1 and C2 in C domain, and D1 and D2 in D domain), all of them belonging to a multicast session. In the example, two tunnel RSVP sessions were created. The first session is placed between the R_{entry} router and the R_{exit1} router and the second session is placed between the R_{entry} router and R_{exit2} router. Each tunnel session provides resource reservation to the end-to-end multicast RSVP session.

When the R_{entry} router receives a *Path* message, it creates the path state of the end-to-end RSVP session. It then verifies through which exit router or exit routers the encapsulated *Path* message will be sent by using the Border Gateway Protocol (BGP) [18]. After choosing the exit router(s), the R_{entry} router sends an encapsulated end-to-end *Path* message to the R_{exit1} router and a copy of the encapsulated end-to-end *Path* message to R_{exit2} router.

The process of reservation set-up continues for each tunnel RSVP session through of steps 3-12 describes in Section 3.

The use of staged refresh timers [13] provides confiability through of acknowledgment messages *PathAck* and *ResvAck*. Multicast aggregation mechanism do not use the additional messages for acknowledgments. The role of an initial tunnel PATH message is to prepare for a subsequent tunnel RESV message. A tunnel RESV message can be considered as an acknowledgment for a tunnel PATH message. A tunnel RESV_CONFIRM message represents an acknowledgment for a tunnel RESV message. The staged refresh timer controls the retransmission interval between tunnel PATH messages and tunnel RESV messages of the way explained in Section II. When the exit router sends the tunnel RESV message first, it includes the RESV_CONFIRM object in message and it includes the object in a waiting list. The RESV_CONFIRM object is composed of the entry router address and of the object number. There is a staged refresh timer associated with every object and the timer controls the retransmission of messages. If the exit router receives the RESV_CONFIRM corresponding, it deletes the object in list, else it retransmites the tunnel RESV message.

Reserved data packets are encapsulated in IP and in UDP protocol in entry router [1]. The use of the UDP encapsulation increases the processing time in routers. The packet marking executed in network layer directly is more fast than in transport layer. Therefore, we propose marking the data packets of the reserved multicast session at the entry router with the traffic class of the TOS field or of the DS field of the IP packet outer header. The DS field is obtained by renaming the TOS field, in the case of the IPv4 protocol, or *Traffic Class* field in the case of the IPv6 protocol [19]. The utilization of the DS field is to maintain compatibility with QoS requests that it uses Differentiated Services. After the marking, the entry router generates copies (one for each tunnel), and every packet is encapsulated and sent to the exit router address. Each exit router receives the packets, decapsulates and sends them to their destination.

As the number of exit routers to where the reserved packet must be sent increases, it also increases the number of packet copies in the region. Consequently, the bandwidth consumed by the network is increased. A solution to this problem is to use the multicast IP address in the destination address field of the outer header of the reserved packet. The reserved multicast address range would be used in aggregation region (e.g., a VPN or a ISP) and each reserved multicast address would represent a group of exit routers. Each router in the region must have a table (multicast address, list of exit routers). The state of each tunnel session would be maintained. The entry router would send a reserved packet and its duplicating would happen only in the router that has two interfaces (Figure 3).

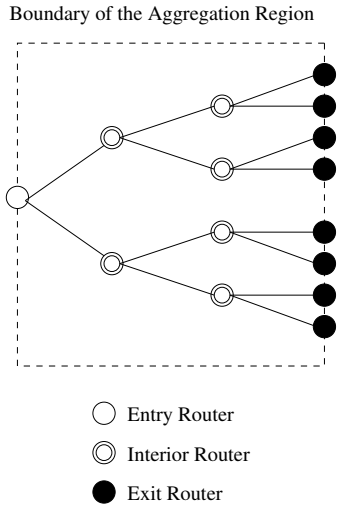

Fig. 4. Simulation Topology

V. Results

This section presents the simulation results of the process of reservation set-up executed by multicast aggregation mechanism when there are tunnel RSVP message losses.

The aim of simulations is to compare the performance between the utilization of staged refresh timers and fixed refresh timers in the aggregation mechanism in presence of tunnel RSVP message losses. The fixed refresh timer refers to classical soft-state mechanism which the retransmission interval is a fixed value (30 seconds) during the flow establishment.

In order to obtain the measurements we have used the *ns* (Network Simulator) [20]. An extension to the *ns* basic packet was implemented to support the aggregation mechanism being formed by two modules: the classic RSVP and the aggregated RSVP. The classic RSVP module was obtained from [21]. The two modules were implemented using the C++ and the Otcl languages.

The aggregated RSVP module developed executes the following functions:

- Processing of tunnel PATH, tunnel RESV, tunnel RESV_CONFIRM and tunnel RESV_ERROR messages.
- Mapping of end-to-end sessions to tunnel sessions.
- Packets encapsulating and decapsulating.
- Aggregated states tables maintenance.
- Retransmission of messages within staged refresh timers.

The simulations consisted of repeated reservation establishments for multicast groups (a group is composed of a source and n receivers) in the aggregation region, over multicast trees of various lengths and under different loss conditions.

Figure 4 presents the topology used in the simulations. A full binary multicast tree where the root node is the entry router and the leaf nodes are the exit routers. The configurations of the multicast tree comprise 3, 7 and 15 nodes.

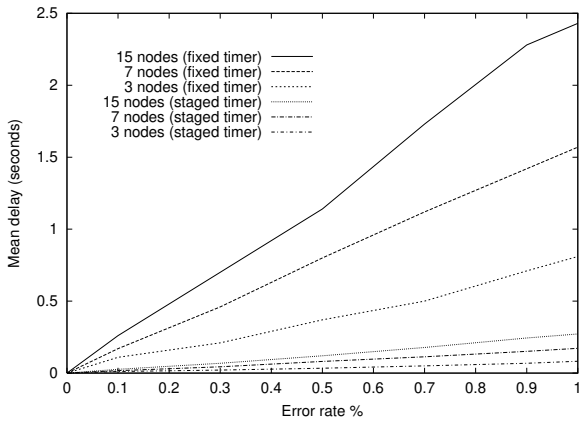

Fig. 5. Mean establishment delay in multicast tree within 0.1-1% of packet loss

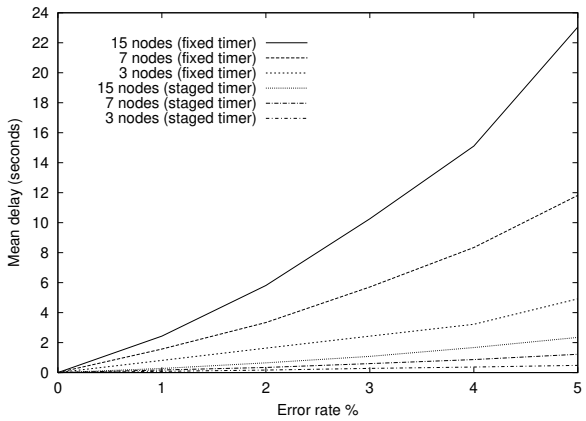

Fig. 6. Mean establishment delay in multicast tree within 1-5% of packet loss

For every configuration, 3000 flows were established and no delay was introduced in links to isolate the time overhead introduced only by the operation of the mechanism. The mean duration of each multicast session and of the transmission between tunnel PATH messages are 150s and 100ms respectively with exponential distributions. A tunnel RESV message is sent immediately after the arrive of the tunnel PATH message.

In simulations of the full binary multicast tree (Figure 4), the loss process on each direction of a link is represented by a packet loss rate with uniform distribution. The links used in simulations have a low error rate (0.1-1%) or a medium error rate (2-5%).

The Figures 5 and 6 show the measured mean establishment delay and clearly demonstrate the gain in performance obtained with staged refresh timers. The error rate represents the packet loss in all links for each configuration of the multicast tree. Note the low establishment delay achieved by staged refresh timers over multicast trees with error rate between 3% and 5% (Figure 6). This result is important in the case of a change of route, since local repairs produce bursts of control messages which can temporarily congest the link.

VI. Final Considerations

The flow aggregation mechanisms provide a decrease in the number of states necessary to store about the resources reservation of RSVP protocol, thus increasing the scalability of its use in the Internet.

The multicast aggregation mechanism proposed in this paper reduces the number of stored states and the number of messages processed in the aggregation region because the interior routers handle only tunnel RSVP messages generated in tunnel end points.

The simulations evaluate the mechanism performance in process of reservation set-up when losses of the tunnel RSVP messages occur. The results shown the gain in mean establishment delay obtained with the use of staged refresh timers over fixed refresh timers. The results also revealed a low establishment delay in multicast trees with error rate between 3% and 5%. This result is important in the case of a change of route, since the router produces bursts of control messages which can temporarily congest the link.

For future works, we can emphasize, the execution of simulations in larger topologies to evaluate the RSVP messages overload associated with the use of staged refresh timers and with the packet encapsulating at tunnel end points.

VII. References

[1] A. Terzis, L. Zhang, and Ellen L. Hahne, "Reservations for Aggregate Traffic: Experiences from an RSVP Tunnels Implementations", *6th International Workshop on Quality of Service (IWQoS'98)*, May 1998.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", *RFC 1633*, July 1994.

[3] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: a new Resource ReserVation Protocol", *IEEE Network Magazine*, September 1993.

[4] J. Wroclawski, "The Use of RSVP with Integrated Services", *RFC 2210*, September 1997.

[5] U. Roedig, C. M. Gortz, M. Karsten, and R. Steinmetz, "RSVP as Firewall Signalling Protocol", *Proceedings of the 6th Symposium on Computers and Communications*, July 2001.

[6] D. Awduche, L. Berger, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", *RFC 3209*, July 2001.

[7] Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang, "A Two-Tier Resource Management Model for the Internet", *Proceedings of Global Internet Conference*, December 1999.

[8] Appan Ponnappan, Lingjia Yang, Radhakrishna Pillai, and Peter Braun, "A Policy Based QoS Management Systems for the IntServ/DiffServ Based Internet", *Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, June 2002.

[9] O. Schelén and S. Pink, "Resource Sharing in Advance Reservation Agents", *Journal of High Speed Networks: Special issue on Multimedia Networking*, vol. 7, no. 3-4, 1998.

[10] Jens Schmitt, Martin Karsten, Lars Wolf, and Ralf Steinmetz, "Aggregation of Guaranteed Service Flows", *7th International Workshop on Quality of Service (IWQoS'99)*, June 1999.

[11] Fred Baker, Carol Iturralde, Francois Le Faucheur, and Bruce Davie, "Aggregation of RSVP for IPv4 and IPv6 Reservations", *RFC 3175*, September 2001.

[12] Prashant R. Chandra, Peter Steenkiste, and Allan Fisher, "Extensible Signaling for Temporal Resource Sharing", *IEEE Journal Selected Area in Communications, Vol. 19 No. 3*, March 2001.

[13] Ping Pan and Henning Schulzrinne, "Staged Refresh Timers for RSVP", *Proceedings of Global Internet Conference*, November 1997.

[14] R. Braden, L. Zhang, S. Berson, and S. Jamin, "Resource Reservation Protocol (RSVP) - Version 1 functional specification", *RFC 2205*, September 1997.

[15] Maya Yajnik, Jim Kurose, and Dom Towsley, "Packet Loss Correlation in the Mbone Multicast Network", *Proceedings of Global Internet Conference*, November 1996.

[16] Charles Perkins, "IP Encapsulation within IP", *RFC 2003*, October 1996.

[17] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language - User Guide*, Addison-Wesley, 1999.

[18] T. Bates, R. Chandra, D. Katz, and Y. Rekhter, "Multiprotocol Extensions for BGP-4", *Internet-Draft*, January 1998.

[19] K. Nichols, S. Blake, F. Baker, and D. L. Black, "Definition of the differentiated services field (DS field) in the IPv4 and IPv6 headers", *RFC 2474*, December 1998.

[20] NS, "Network Simulator - NS (version2)", *http://www-mash.cs.berkeley.edu/ns/*, 2000.

[21] Marc Greis, "RSVP/ns: An Implementation of RSVP for the Network Simulator ns-2", *http://titan.cs.uni-bonn.de/ greis/rsvpns/rsvpns.ps.gz*, 1999.