Performance Evaluation of the Scalability Effects on Control-Plane Signaling in Resource Reservation Protocol (RSVP)

Rajiv Chakravorty *
Multimedia Technology Group
Silicon Automation Systems Ltd.
HAL III Stage, Bangalore - India
Email: rajivc@sasi.com

Subrat Kar
Electrical Engineering Dept.
Indian Institute of Technology,
New Delhi - India
Email: subrat@ee.iitd.ernet.in

Peyman Farjami
Chair of Communication Networks
Aachen University of Technology,
Aachen - Germany
Email: pemi@comnets.rwth-aachen.de

ABSTRACT

This paper evaluates the scalability of control-plane signaling in Resource Reservation Protocol (RSVP). RSVP has been developed to support resource reservation upon request by generating explicit control messages for real-time sessions demanding QoS. It maintains its soft state for these sessions by periodically refreshing the already established session states. SDL(Specification and Description Language) was used for modeling purposes. Load generators generating both real-time as well as non-real time traffic sessions were used during the evaluation. A number of sessions were set-up, and performance evaluation carried out by increasing the number of sessions. The load on the control-plane was observed by varying the number and type of sessions.

The paper attempts to analyse IP-signaling protocols meant for reservations. We believe that such an evaluation provides a reasonable picture on the control-plane overhead related to signaling-protocols like RSVP. The paper also demonstrates the use of load generators to model a real network (traffic) scenario necessary for such an evaluation.

1 Introduction

IntServ IP QoS architecture provides all the benefits of a QoS enabled network. RSVP (resource reservation protocol) is the working protocol for IntServ. It is a signaling protocol where all participating end-systems establish a closed control loop. The protocol provisions resources (bandwidth) at each of the network elements (routers) along the route. Developers of RSVP have embedded it with lots of useful features. Few of them include: support for unicast/multicast, receiver-initiated reservations and soft-state adaptiveness.

However, widespread deployment of RSVP in the Internet, as thought of a few years back, has not materialized. This can be attributed to certain shortcomings associated with RSVP. First, it involves costly and complex signaling in the Internet. The periodic control messages of RSVP is an additional burden, eating up an additional chunk of the network bandwidth. Secondly, it is difficult to anticipate the exact nature of certain streams. This may sometimes lead to over-reserving of resources and consequently leading to lower network utilization. In addition, bandwidth reservation in the Internet requires new charging and settlement mechanisms. Even if all of these were accountable, the final problem that should concern RSVP is its scalability. It is said that RSVP does not scale with large number of sessions. It is this aspect of RSVP that makes it practically infeasible towards its deployment in the Internet, especially in the backbone.

In this paper, we intent to analyse the scalability of the protocol at the control (signaling) plane. Here, our goal is to quantify the scalability of processing signaling messages (i.e. load on control plane) in RSVP as a function of:

1. Total number of concurrent reservation sessions.

2. Total number of real-time (guaranteed service sessions) to non-real time (best-effort) sessions.

3. Type of the real-time(e.g. speech) and non-real (e.g. ftp) traffic.

4. Intensity of any new reservation attempts.

*Rajiv was a recipient of DAAD Scholarship (Germany) during his Masters project at Aachen University of Technology, Aachen, Germany and IIT-Delhi, India

The performance at the control-plane for the QoS signaling in case of RSVP needs evaluation. This is because:

1. It directly affects the end-user applications (e.g. reservation setup time).
2. It concerns the problems that network designers are facing (e.g. maximum number of reservations).
3. Heavy signaling can have a serious impact on handling of other protocols (e.g. routing, management - if they are not isolated).
4. Can affect traffic engineering related issues.

It is important to note here that the traffic-handling approach (per-flow) in RSVP is not directly coupled to signaling. Thus, the data-plane processing overhead due to conditioning of the traffic of reserved flows is also independent from the performance of the signaling protocol. In this paper, we have concentrated on the scalability effects of RSVP signaling *with* per-flow and per-flow-type of traffic handling, as opposed to just RSVP signaling.

Apart from those discussed above, we also intend to extend this study to a prototype simulation set-up for RSVP that will help us to evaluate the feasibility of IP signaling protocols used for QoS (such as RSVP) provisioning in the Internet. There has been some field trails conducted for RSVP with little success [2, 3]. To pursue further with the idea of studying IP-signaling protocols like RSVP extended to a simulation model has been somehow avoided. This can be mainly due to non-availability of right kind of load generators to generate different traffic types, required for such an evaluation. In this paper, we will show the use of load generators that generates various kinds of traffic, with the RSVP model developed in SDL. Later, we will discuss the results of the performance evaluation.

The paper is organized as follows: The next section gives a brief overview on RSVP. Section 3 details the control plane signaling involved in RSVP. Section 4 decribes the modeling of the system in SDL. Here, a detailed explaination for use of SDL along with issues like receiver-end bandwidth computation model, admission control and traffic scheduling that have been modelled in SDL is given. This additionally includes the description on the SPEET Class Library, load generators used and set-up scenario for the performance evaluation. Section 5 discusses the results of the performance evaluation. And finally, conclusions are drawn in Section 6.

2 Overview of IntServ-RSVP

Integrated Services (IntServ) architecture assumes that resources are reserved at each router along the path between sender and receiver using explicit signaling (using RSVP) for any flow that demands QoS. IntServ has specified three types of services: *Guaranteed Quality Service*, *Controlled Load Service* and *Best Effort Service* [8, 9].

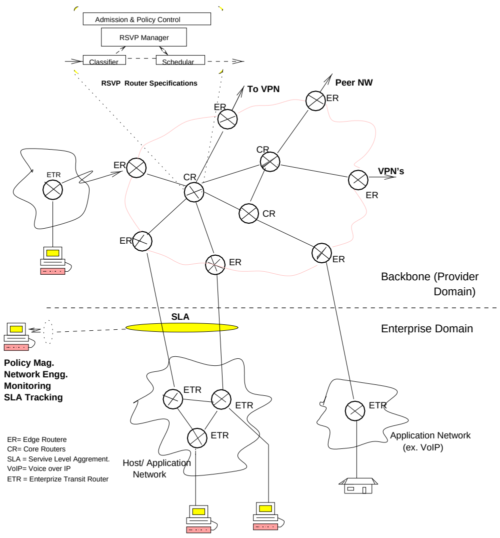

Figure 1: IntServ IP-QoS Model

Path messages go upstream all the way from sender to receiver while the receiver returns a *Resv* message for each *Path* message sent along the same route. Each router along the route stores the state for each such session flow established. The receiver receives the *Path* message that provides it with the information of the application and all details pertaining to that route (like end-end delay, available bandwidth etc.). With this information available, the receiver then computes the required bandwidth and sends this request in a *Resv* message to all the routers along the route to reserve bandwidth. Based on availability, bandwidth is reserved in the routers. When this message arrives back to the sender, the reservation is considered successful. Any error during the connection set-up process will trigger error messages that removes the states established for a given session along the route. It will then try to re-establish the reservation from an alternate path using the underlying routing protocol. This makes RSVP a soft-state adaptive protocol.

3 Control Plane Signaling in RSVP

In RSVP, the periodic *Path* and *Resv* messages are responsible for maintaining states in the router. However, these messages can be lost in the network, mainly due to congestion [7]. To maintain the existing states, Timeout duration (T) in a router is adjusted such that some messages are lost before the actual Timeout occurs. Therefore, to account for the Timeout interval's dependence upon the extent of the network congestion,the Timeout interval can heuristically [9] governed by:

$$T = (K + 0.5) \times R \times 1.5$$

where R is the Refresh Interval with default value of 30 seconds. The congestion dependent constant K (which is usually a small integer depending upon network congestion) indicates that $K - 1$ messages will be lost before the actual Timeout should occur. This *soft-state* adaptive nature provides fault tolerance to the protocol. Similarly, the value of R is set to a random value in the range of $\{0.5\text{R}-1.5\text{R}\}$ in the actual Internet to avoid TCP Global message synchronization. As shown in Figure 2, there are n network

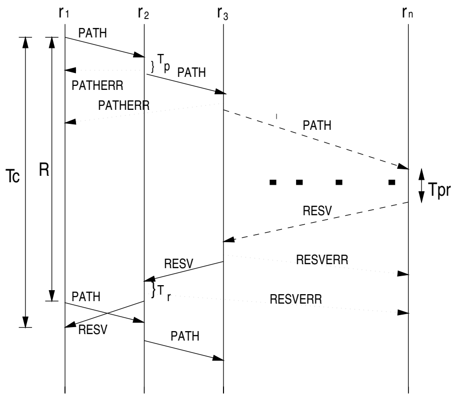

Figure 2: Control Plane Signaling in RSVP

elements (all RSVP capable) $r_1, r_2, r_3 \ldots r_n$ along a given path. The sender (r_1) sends the *Path* message periodically at a given refresh rate. The receiver (r_n) receives this message that has two important information i.e.*Tspec* - traffic specification of the application and *Adspec* - advertising specification indicating the resources supported along the route. Using this information the receiver computes an optimum bandwidth using the computation model which is again requested from the routers along the path. If we neglect the propagation and transmission delay, the connection set-up time for a given session is given [3] by:

$$T_c = (T_p + T_r) \times n \times T_{pr}$$

where n is number of routers along the path, T_p is average *Path* message computation time in a router, T_r is the average *Resv* message computation time and T_{pr} is the average receiver-end bandwidth computation time.

Consider a case for a router that has crashed along the path of an existing session with no chance of an early recovery. In such a case, all the routers along that route will timeout before the sender can restore the reservation states from an alternate route. If the crashed router had recovered before a single refresh interval, the neighbouring routers would have communicated the states to the crashed router. As this is not the case, the reconnection set-up interval (T_{rec}) will be the sum of Timeout interval (T) and connection set-up time (T_c).

$$T_{rec} = T + T_c$$

It is for this duration the end-user is deprived of the desired QoS, before the reservation can restored along an alternate path. Moreover, the reconnection set-up interval depends upon Timeout interval. Timeout interval is a function of the congestion dependent constant K and on refresh interval R. Changing any of these values to get an optimum value of T is difficult. Reducing value of R will lead to increased network overhead. An improved method has been proposed [7] with staged refresh timers that leads to dynamic optimization of refresh-interval R. In the same way, reducing constant K below the default value will lead to *soft-state* instability and more frequent route timeouts. This is more common in links, prone to congestion.

On the other hand, if propogation and transmission delay is neglected, the connection set-up time T_c depends upon processing overhead for $\{T_p, T_r, T_{pr}\}$ of the RSVP signaling messages. It is obvious that the values $\{T_p, T_r, T_{pr}\}$ are influenced by the extent to which a router is loaded. So, as the computational burden in a router increases, these values are also expected to increase. This paper is specifically meant to evaluate the processing overhead for these control messages with respect to load on the data-plane.

4 Modeling Strategy

The complete RSVP model was developed using the graphical representation of Telelogic's SDL v3.4 (Specification and Description Language). SDL also produces a phrasal representation (.pr files). This phrasal representation was then appropriately converted into C++ code using a SDL-to-C++ translator. Any system built using SDL can map ADT's

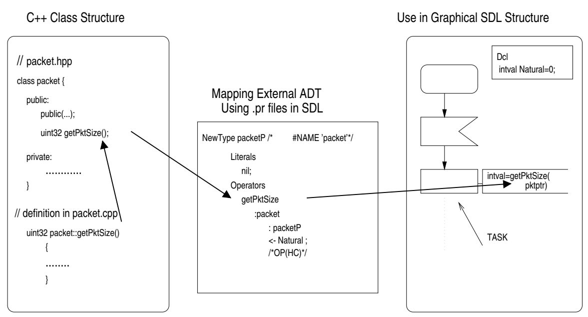

Figure 3: Mapping external ADT's in SDL

(abstract data types) using external class library. Figure 3 shows external C++ classes being mapped into the SDL specification. We extensively used SPEET

(SDT Performance Evaluation Tool) Class Library, developed in-house at COMNETS [13, 14]. It features generic C++ classes as well as a simulation library with its strength concentrated more on random number generation, statistics and event driven simulation. All the blocks as described in [9] and others specifications mentioned in [10, 11, 12] about RSVP were modeled using the graphical representation of SDL. The generic container classes and event driven classes (e.g. *EventHandler* class) of SPEET were used to build the system. However, policy control block was not modeled. One of the main blocks (admission control) modeled using Telelogic's SDL is shown in Figure 4. Our intension was to generate explicit

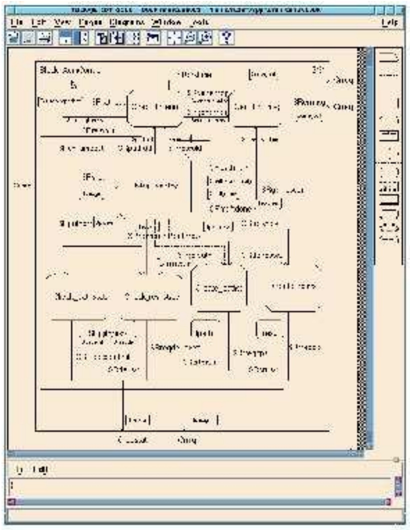

Figure 4: SDL block for Admission control

signaling messages for each of these sessions to establish the desired session states. Once the states were successfully established, load generators generated the required data packets for the respective sessions (here speech, FTP, SMTP and TELNET). The data packets so generated were scheduled according to the respective scheduling strategy for each of the sessions.

Signaling messages for RSVP were generated from the environment. Figure 5 shows the RSVP structure as a part of the SDL specification while the control message generator as part of the environment. There are two types of event-handler classes (i.e. inherited from the *EventHandler* class that are used to generate events (here packets). One of them is used for generating data packets while the other is used for generating RSVP control messages. Here, event handler class represents finite state machines. Initially, number of sessions to be used in the evaluation is obtained, and correspondingly an equal number of load generator, receiver and control message generator objects are instanciated. The RSVP message generator generates periodic *Path* messages that subscribes to the receiver and computes desired bandwidth according to a com-

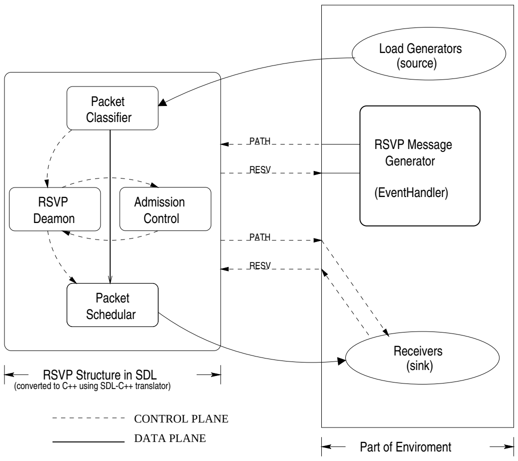

Figure 5: SDL Structure and its Interaction with the Environment for a given session

putation model (only for guaranteed services session) and requests an optimum bandwidth back from the router. The environment (again RSVP message generator) receives this message that again keeps track of periodically refreshing the already established states in the router. Once each of the sessions states are established, the other event-handler (load-generator) then instanciates new data packets and attaches the payload depending upon the types of session event received from the respective load-generator event handlers. These packets are then sent to the environment, which are then transferred to the SDL router structure.

4.1 Receiver-End Bandwidth Computation Model For Guaranteed Services

For the receivers receiving guaranteed service sessions, an optimum bandwidth needs to be requested by the receivers from the network elements (routers) along the path. The calculation of the bandwidth is accomplished by considering the token bucket specification for the given sessions. The following describes the receiver-end bandwidth computational model in brief.

The delay bound for an application with token bucket specification of depth b and rate r is computed to $\frac{b}{R}$, provided $R \geq r$ where R is the link bandwidth. For guaranteed services, the contribution by the schedular have to taken into account at each node (router) along the route. The *error terms* $\{C_{tot}, D_{tot}\}$ represent the extent to which the network element's implementation for guaranteed service deviates from the fluid model. If packetization effect is taken into account (with maximum size to be M), the resultant end-end delay in such a case is given by [1, 8, 12]:

$$\hat{D} = \left\{ \begin{array}{ll} \frac{(b-M)(p-R)}{R(p-r)} + \frac{(M+C_{tot})}{R} + D_{tot} & (\quad p > R \geq r) \\ \frac{(M+C_{tot})}{R} + D_{tot} & (\quad R \geq p \geq r) \end{array} \right\}$$

where, C_{tot}, D_{tot} is the cumulative sum of the error $\{C_{tot} = \sum_{i \epsilon n} C_i, D_{tot} = \sum_{i \epsilon n} D_i\}$ introduced at each of the node along the end-end data path. C_{tot} is the rate-dependent error term that represents the delay a datagram in the flow might experience due to the rate parameters of the flow. The error term D_{tot} is the rate-independent, per-element error term and represents the worst case non-rate-based transit time variation through the service element [12].

Receiver extracts the *Tspec* and *Adspec* objects from the *Path* message. It computes the actual Queuing delay $Q_{QueActual}$ by substracting minimum path latency available from *Adspec* object of *Path* message from the delay specified in Application (obtained from *TSpec* object of the *Path* message). On the other hand, the theoretical delay ($Q_{QueTheory}$) is obtained by substituting $R = p$ (taking peak rate into account) in the equation for end-end delay above. If the value of the theoretical delay $Q_{QueTheory}$) obtained is higher than the actual Queuing delay $Q_{QueActual}$, then R_{min} is calculated using the same equation above with the case when $(R \geq p \geq r)$. Otherwise, R_{min} is obtained by using the equation for the case when $(p > R \geq r)$. The *Rspec* object for the *Resv* message is constructed using the minimum of $\{R_{min}, R_{AdSpec}\}$ value obtained.

4.2 Admission control

Existing and past research have focussed extensively on various admission control algorithms. In an Integrated Services packet network, the choice of any admission control algorithm depends upon:

1. Allowing enough bandwidth fluctuations in Guaranteed Services and Controlled load service classes.
2. Not subjecting the Best effort sessions to impossible delay demands.
3. Low implementation and computation cost along with high degree of bandwidth utilization.
4. No delay bounds violation in any given service class due to the addition of a new flow.

There are several sophisticated admission control algorithms available. *Measurement based* admission control algorithm[18], specifically meant for integrated services network, not only satisfies the first three criteria mentioned above but also ensures that the delay bounds are not violated upon admission of a new flow, even if the new flow displayed worst-case behaviour. However, one of the drawbacks associated with this algorithm is its implementation and operational costs. This becomes a prohibitive factor towards widespread deployment of such an algorithm in the Internet.

The *measured sum* algorithm {[15], see pg. 77 } satisfies the first three conditions. To satisfy the fourth condition one needs to consistently calculate the conservative delay estimates [4]. This ensures that the delay bound of a given service class will not be violated by the admisssion of any new flow. Instead, the *measured sum* algorithm introduces the concept of *utilization target* that ensures availability of some percentage of the bandwidth permanently to the *best effort* sessions. The algorithm performs the following test for admission:

$$\hat{v} + r_\alpha \leq v\mu$$

where r_α bandwidth of any the new flow, v is the total link bandwidth available and μ is the *utilization target* usually in the range (0.8 - 0.9) which the guaranteed service and controlled load sessions utilize while the rest ($10\% - 20\%$) is left for best effort sessions. Moreover, unlike the *measurement based* algorithm, the use of *measured sum* algorithm is simple and leads to reduction of the overall implementational and computational costs.

Even tough the requirements for an admission control algorithm are clearly defined, the need to implement all the them in the present evaluation is not required. A simple algorithm has been considered that should not impose any strict restrictions for admission but instead facilitate easy establishment and admission of the sessions.

4.3 Scheduler Implementation

A framework for implementation of the scheduler is shown in figure 6. The scheduling framework considered both the real-time (guaranteed services) and non-real time sessions (best effort sessions). The choice of an appropriate scheduling discipline is based on the requirements imposed by a given service class. Usually, this depends upon the nature of performance guarantees, efficiency, algorithm complexity (both in terms of implementation and control overhead) and amount of flexibility required for that service class. To pro-

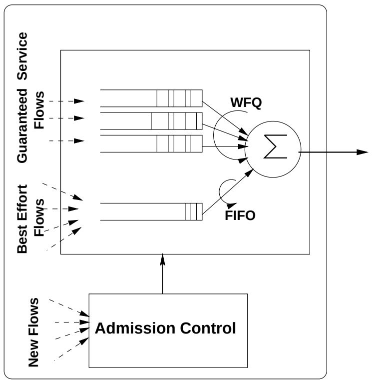

Figure 6: Implementation of the Schedular

vide guaranteed bandwidth for rigid real-time appplications, the scheduling stategy should feature: fair share of bandwidth, isolation of sessions/classes and QoS consistency. Weighted fair queuing (WFQ) has been recommended for such applications [9]. WFQ is a work conserving algorithm that ensures fairness to all sessions at any time instant. The basic idea in WFQ relates to Generalized Processor Sharing (GPS) where a weight ϕ_i is associated with flow $i, i = 1, \ldots, N$ and the link capacity shared among the flows is in direct proportion to their weights. In other words, if γ were to denote the link speed, then flow i is guaranteed to obtain a minimum service rate of $\frac{\phi_i}{\sum_{j=1}^{N} \phi_j}\gamma, i = 1, \ldots, N$. However, in reality, GPS represents a fluid (reference) model that is replaced by a more practical Packetized GPS (PGPS)- also termed as WFQ. To account for the cumulative impact of the packetized approach for the scheduler (that does not exceed the transmission time of a maximum size packet in terms of delay), the error terms as discussed in section 4.1 are used. Thus, for any token bucket flow having an assured rate (as provided by WFQ) through a given network, has its overall delay bounded [1]. A more detailed explanation of use of WFQ and implementation of WFQ in particular is given in [1, 5, 6].

On the other hand, the best effort sessions are treated with the normal FCFS (FIFO) queuing. This type of scheduling strategy does not provide any delay or rate guarantees. As indicated in earlier subsection, admission control is based upon allocating ($10\% - 20\%$) of the network bandwidth (depending upon value of μ) parmanently to best-effort sessions. Rest of the available bandwidth is used by guaranteed services flows or can be used by best-effort sessions (depending upon availability). Note here that we have not considered the use of controlled load services for the performance evaluation.

4.4 Use of Load Generators

Traffic load generators for individual traffic and mixed traffic types were used. These traffic load generators are part of the SPEET Class Library [14] developed at COMNETS to assist in the evaluation and verification of various communication protocols. SPEET discerns OSI sublayer independent and OSI sublayer dependent applications. The packet sizes of IP oriented services such as FTP, and SMTP may vary in a wide range from some bytes to several mega bytes. Packets of bigger sizes need to be segmented depending upon the sublayer flow control protocols. On the other hand, OSI sublayer independent applications either depend on the user only, e.g. the speech and telnet generators, packet sizes of such applications are relatively small and, due to this, there is no need to segment these packets in sublayers, which expect the respective sublayers to sustain a certain bit-rate and ignore sublayer flow control mechanisms. The speech generator models talkspurts and pauses of one speaker in a conversation as proposed by Brady [16]. The traffic load generators of IP oriented services are based on long-term measurements [17].

The model parameters that characterize a session of an application are: interarrival time, session duration, total size of transmitted data, distribution and duration of phases of activity and inactivity, number of transmitted objects per session, distribution of object sizes, direction of transmission (bidirectional or unidirectional), and destination addresses of connections. In the present scenario, the load generator objects basically allow three important parameters to specify a session: inter-arrival time, total size of transmitted data and session duration.

4.5 Set-up Scenario

Evaluation was carried out with guaranteed service class sessions (real-time), best-effort (non real-time) sessions and a mix of both types of service class sessions. FTP traffic was selected to represent non-real time traffic type corresponding to the best effort service that is usually the case in a network involving large data transfer (files). For real time session (guaranteed services), we used a load generator for conversational speech (audio) session. All load generators were initialized for generating session for infinite time duration. The inter-arrival time of the generated data packets, their sizes and session duration for each of the load generator instances were initialized using the respective load generator objects. The provision to generate a session with a specified inter-arrival time and packet size determines the overall rate of the session.

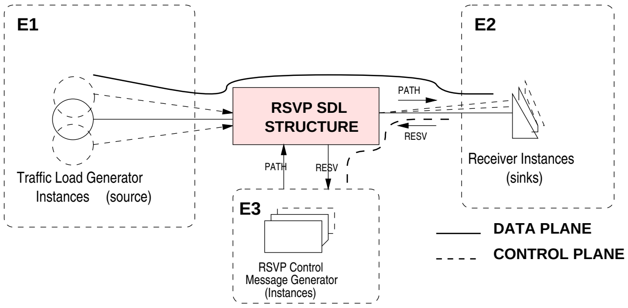

Figure 7: Set-up for Evaluation

Figure 7 shows the set-up for the simulation model. The load generator instances (E1), receiver instances (E2) and RSVP signaling module (E3) are part of the environment. Initially, *Path* messages are sent corresponding to each individual session. Each session is identified by a unique session identifier (sess-id). The states associated with these messages for a given session are then mapped onto the *Path* states for the

RSVP SDL structure. Once the request for reservation of a given session is successful (after *Resv* message was received), the load generator instance corresponding to that session is indicated, to start generating data. The periodic messaging in the mean time continues for each of the established session.

Three different cases for the performance evaluation were considered. In the first case, only non-real time sessions (here FTP) while in the second case, only real time sessions using the conversational speech (audio) generator were used. The last case modelled mixed traffic types that included speech, SMTP, TELNET and FTP. In all the three cases, receiver-end bandwidth computation time was calculated for a given receiver instance. However, in the final case we measured the parameters T_p, T_r, T_{pr} simultaneously for speech (real-time) session and a FTP (non real-time) session respectively. The connection set-up time (T_c) for a given session was then obtained by the summation of all the values of *Path* and *Resv* message processing time (T_p and T_r) and receiver-end bandwidth computation overhead (T_{pr}) for that session. Averaged values of the first 100 control-messages were considered.

5 Performance Evaluation

Performance evaluation was carried out in a SUN (ultra-1, OS v5.6) workstation. Sessions were set up by instantiating equal number of receivers and load-generators. Each generator instance was linked to provide a unique session to a given receiver instance. The load generator instances acted as source for the data while the receiver instances acted as sinks.

In the first case involving FTP sessions, the number of sessions were set to a fixed value of 10. This was increased to 50 and then in steps of 50 to reach to a maximum target of 250 FTP (non-real time) sessions. For a given FTP session, the readings for the *Path* message computation time (T_p), *Resv* message computation time (T_r) and receiver-end bandwidth computation time T_{pr} were averaged to get a final weighted average values for $\{T_p$, T_r and $T_{pr}\}$respectively. Their values obtained for different number of FTP sessions are plotted in Figure 8. The connection set-up time (T_c) is the summation of all the three i.e. T_p, T_r, T_{pr}.

For the second case, a similar measurement meant for real time sessions (speech-audio) was also carried out, this time by instantiating the speech (audio) generators to generate the desired number of sessions. The readings obtained for a given speech session was averaged. The plotted result is shown in Figure 9. The last case involved mixed traffic sessions. We instantiated different number of sessions of each of the given type (speech, FTP, SMTP and TELNET) each time and performed a similar measurements as indicated earlier. Values of T_p, T_r and T_{pr} were obtained by

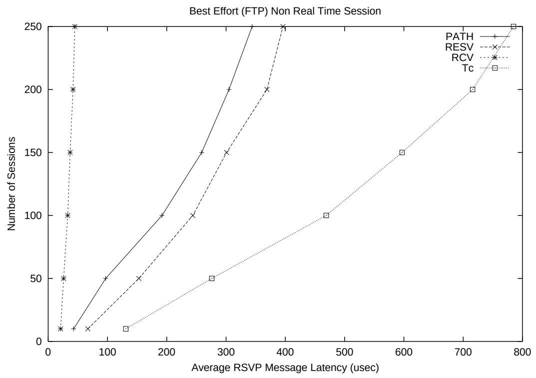

Figure 8: T_p, T_r, T_{pr} and T_c for a FTP session

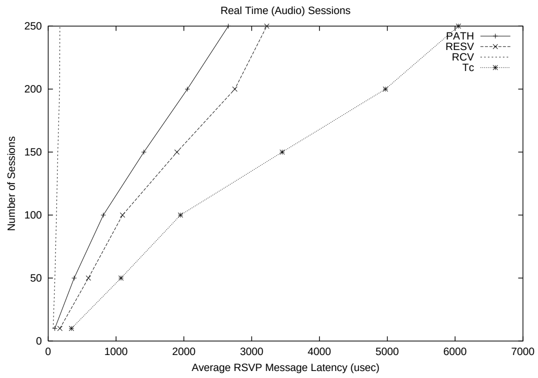

Figure 9: T_p, T_r, T_{pr} and T_c for a speech session

increasing proportionally each of the generator types, but this time reaching a higher value for a maximum of total 450 sessions. The composition of different generator instances is given in Table 1. As indicated earlier, the measurement of T_p, T_r, T_{pr} was simultaneously measured for two seperate cases:

1. Receiver receiving a best-effort service (non real-time) session (Ftp session, see Figure 10).

2. Receiver receiving a guaranteed service (real-time) session (speech session, see Figure 11).

Measurements were carried out for both these cases; the values of $\{T_p, T_r, T_{pr}$ and $T_c\}$ for each of these cases are plotted in Figure 10 and Figure 11 respectively. The numerical values are shown in Table 2 and Table 3. Table values are in milli-seconds whereas the figure values are indicated in micro-seconds.

The driving point for the third case was to evaluate the computation overhead i.e. $\{T_p, T_r, T_{pr}\}$ for a guaranteed service session (speech) and a best effort session (here ftp) simultaneously in the presence of mixed traffic. Upon close observation from Table 2 and Table 3, we find that all the parameters, i.e. T_r, T_p and T_{pr} are slightly more for the speech (guaranteed service) session as compared to that of a ftp (best-effort) session.

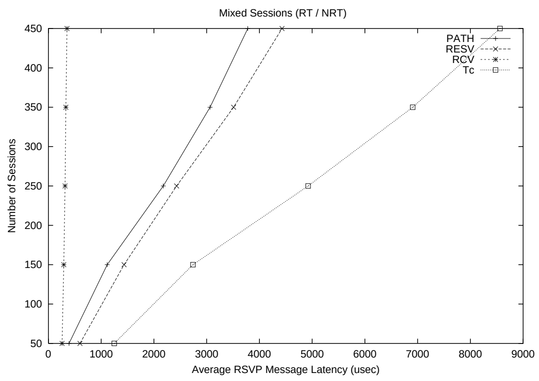

Figure 10: T_p, T_r, T_{pr} and T_c for mixed traffic with a FTP session (refer table 2)

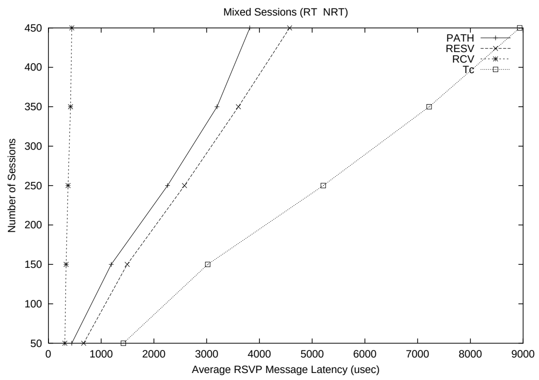

Figure 11: T_p, T_r, T_{pr} and T_c for mixed traffic with a speech session (refer table 3)

6 Observations and Conclusions

We performance evaluated RSVP along the control-plane in presence of different types of network traffic. The motivation for such an evaluation was to observe the effect of scalability along the control-plane in RSVP i.e. effect on control message processing time T_p, T_r, T_{pr} with increase in number and type of sessions states. From the results, it is obvious that with the increase in number of sessions, the control message processing time (T_p, T_r) also increased. Similarly, a slight increase in receiver-end bandwidth computation time (T_{pr}) with the increase in number of sessions was observed. Other observations include:

Sessions	VoIP	FTP	SMTP	Telnet
50	20	15	10	5
150	60	40	30	20
250	100	70	50	30
350	150	100	60	40
450	200	120	80	50

Table 1: Mixed traffic (session) distribution

Sessions	T_p	T_r	T_{pr}	T_c
50	0.391	0.598	0.259	1.248
150	1.115	1.434	0.292	2.741
250	2.177	2.431	0.315	4.923
350	3.065	3.511	0.331	6.907
450	3.778	4.449	0.353	8.561

Table 2: T_p, T_r, T_{pr} and T_c for mixed traffic with a ftp session

Sessions	T_p	T_r	T_{pr}	T_c
50	0.443	0.668	0.311	1.422
150	1.191	1.494	0.335	3.020
250	2.258	2.580	0.372	5.210
350	3.197	3.601	0.419	7.217
450	3.818	4.573	0.444	8.935

Table 3: T_p, T_r, T_{pr} and T_c for mixed traffic with a speech session

1. A higher *Resv* message processing time (T_r) than the *Path* message processing overhead (T_p). This is because the *Resv* message has to bear the additional overhead to perform an additional admission control tasks for a given session.

2. The *Path* and *Resv* message processing overhead is higher for a guaranteed service session than for a best-effort session. The computation time (T_{pr}) is also higher because of the additional receiver-end bandwidth computational overhead associated with the guaranteed service sessions.

From the second observation, it is reasonable to conclude that the connection set-up time T_c is always higher for guaranteed service (real-time) sessions as compared to best-effort service (non real-time) sessions.

The results presented above provides a fair picture on control-plane overhead associated with signaling protocols like RSVP in the Internet. Since modeling a real network scenario is not an easy task, a simulation environment that is very similar (but not exactly) to that of a real one has been attempted. A similar kind of evaluation can also be performed for any other protocols before deploying them and carrying out their field trials in the Internet.

A major issue in this paper that has not been addressed is the performance evaluation for the controlled load service sessions. As such, we were limited only to real-time (guaranteed service) sessions and non real-time (best-effort) sessions.

Acknowledgements

The authors would like to thank Prof. B. Walke of Chair of Communication Networks (COMNETS), Aachen University of Technology, Germany for providing the necessary motivation for this project and also for making the tools available (SDL, SPEET Class library and SDL-to-SPEETCL translator) during the project.

References

[1] R. Guerin, V. Peris, "Quality of Service in a packet networks: basic mechanisms and directions", *Computer Networks*, ELSEVIER, Vol. 31, pp. 169-189, 1999.

[2] Mark Baugher, Saib Jarrar, "Test results of the commercial Internet Multimedia Trials", *Computer Commmunications Review*,ACM SIGCOMM, pp. 16-27, 1998.

[3] T. Chiueh,A. Neogi, Paul Stirpe, "Performance Analysis of a RSVP-Capable Router", *IEEE Real Time Technology and Symposium*, Denver, June 1998.

[4] David D. Clark, Scott Shenker and Lixia Zhang "Supporting Real-Time Applications in an Integrated Services Packet Network Architecture and Mechanism", *Proceedings of SIGCOMM* Baltimore,MD, August, 1992. http://ana-www.lcs.mit.edu/anaweb/papers.html

[5] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm", *Journal of Internetworking Research and Experience* , Vol.1, No.1, pp 3-26, September 1990

[6] S. Keshav, "On the Efficient Implementation of Fair Queueing", *Journal of Internetworking* , Research and Experience, Vol.2, No.3, September 1991

[7] Ping Pan, Henning Schulzrinne, "Staged Refresh Timers for RSVP", *Global Internet*, Pheonix, AZ, November 1997.

[8] Paul P. White, "RSVP and Integrated Services in the Internet", *IEEE Communications Magazine*, pp.100-106,May 1997.

[9] R. Braden, L. Zhang, S. Berson, and S. Jamin, "Resource Reservation Protocol (RSVP) Version 1 - Functional Specification", *IETF RFC 2205*, Proposed Standard, September 1997.

[10] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", *IETF RFC 2210*, Standards Track, September 1997

[11] J. Wroclawski, S. Shenker, "General Characterization Parameters for Integrated Service Network Elements", *IETF RFC 2215*, Standards Track, September 1997

[12] Shenker, S., Partridge, C., and R. Guerin "Specification of the Guaranteed Quality of Service", *IETF RFC 2212*, Standards Track, September 1997.

[13] Martin Steppler, "Performance Analysis of Communication Systems Formally Specified in SDL", *In Proceedings of The First International Workshop on Simulation and Performance '98 (WOSP '98)*, Santa Fe, New Mexico, USA, 1998.

[14] Formal specification and performance evaluation of communication systems *SPEET Class Library,* SDT Performance Evaluation Tool Class Library ,http://www.aixcom.com (downloadable version available).

[15] Sugih Jamin, "A measurement-based Admission Control Algorithms for Integrated Services Packet Networks", *Phd. Thesis*, Univeristy of Southern California, SC Techincal Report, USC-CS-96-639, 1996.

[16] Paul T. Brady, "A Model for Generating On-Off Speech Patterns in Two-Way Conversation", *The Bell System Technical Journal*, Vol. 48, No. 9, pp. 2445-2472, September 1969.

[17] Vern Paxson, "Empirically-Derived Analytic Models of Wide-Area TCP Connections", *IEEE/ACM Transactions on Networking*,Vol. 2, No. 4, pp. 316-336, August 1994.

[18] Sugih Jamin,Peter Danzig, Scott Shenker, and Lixia Zhang, "A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks", IEEE/ACM Transactions on Networking, 5(1):56-70. Feb. 1997