IP TRAFFIC MODELING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY

SİBEL TARIYAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2007

Title of the Thesis : **IP Traffic Modeling**

Submitted by **Sibel Tarıyan**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

Prof. Dr. Yurdahan Güler
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Mehmet Reşit Tolun
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Reza Hassanpour
Supervisor

Examination Date : 08.02.2007

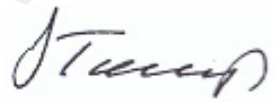
Examining Committee Members

Asst. Prof. Dr. Bülent Tavlı	(TOBB Univ.)	
Assist. Prof. Dr. Reza Hassanpour	(Çankaya Univ.)	
Dr. Abdülkadir Görür	(Çankaya Univ.)	

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Sibel Tarıyan

Signature : 

Date : 08.02.2007

ABSTRACT

IP TRAFFIC MODELING

Tarıyan, Sibel

M.S.c., Department of Computer Engineering

Supervisor: Assist. Prof. Dr. Reza Hassanpour

February 2007, 77 pages

RSVP allows Internet real-time applications to request a specific end-to-end QoS for data stream before they start transmitting data. In this report firstly an overview of RSVP is presented. After that the different quality of services available and the relation between QoS and RSVP are explained. Then the fundamentals of RSVP as a protocol are discussed. The performance issues and benchmarking are given next. The experimental results and discussions conclude this thesis.

Keywords: RSVP, Quality of Service (QoS), Data flow.

ÖZ

İNTERNET PROTOKOL TRAFİK MODELLEMESİ

Tarıyan, Sibel

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Yrd. Doç. Dr. Reza Hassanpour

Şubat 2007, 77 sayfa

RSVP internet gerçek zamanlı uygulamarnda veri dizgisi için gönderimden önce bir uçtan diğer uca servis kalitesi isteğinde bulunmaya izin verir. Bu tezde öncelikle kuşbakışı RSVP sunulmuştur. Bunu takiben farklı servis kaliteleri ve servis kalitesi ve RSVP arasındaki ilişki açıklandı. Bir protokol olarak RSVPnin temelleri ele alındı. Performans sorunları ve karşılaştırmalı sınama sonuçları verildi. En son olarak deneyler ve sonuçları ele alındı.

Anahtar Kelimeler: RSVP, Servis Kalitesi, Veri akışı.

ACKNOWLEDGMENTS

The author wishes to express his deepest gratitude to his supervisor Asst. Prof. Dr. Reza Hassanpour and without his guidance and helps this thesis would not have been completed. I would like to extend my thanks to my family. They were always patient and supporting me with love and care.

Roya Choupani deserves a special thank for everything she did throughout my thesis study. She was always helpful and comforting me when I was stuck with my study. I wish to extend my thanks to Mehmet Reşit Tolun, Department of Computer Engineering chairman.

I must also say a special thank you to all my teachers at the Department of Computer Engineering at Çankaya University. Finally, I would like to thank Tansel.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Internet allows the transmission of data between end points. In original design, it tries to transmit as quickly as possible but there is no guarantee to the timeliness and assurance of actual delivery. It provides its best effort service at end points. It may give qualitatively better service, but without the quantitative bounds of a guaranteed service it is far from expectations especially in an environment containing various services to be handled in the media. There is a great deal of interest in network applications. Accomplishment of best effort service for one single service is far from present day constraints. Due to demanding changes in end-point requirements, internet is affected to meet the quality of service (QoS) requirements. There are several protocols for real time services(Video Conferencing, Internet TV and Internet Telephony, are rapidly growing and perfected) that support QoS of multimedia applications for IP networks such as Resource Reservation Protocol(RSVP) , together with Real-Time Transport Protocol(RTP) , Real-Time Control Protocol(RTCP), Real-Time Streaming Protocol(RTSP),provides a working foundation for real-time services.

Utilization of RSVP in a network with different perspectives is analyzed and simulations conducted are given in experiments.

In this thesis, OPNET network simulation tool is used. OPNET is a network simulation tool that outputs the characteristics of a real time network utilizing different services with a priori parameters. Under given architecture and protocol, performance of quality of service implications is carried out.

This thesis contains nine chapters. Chapter 1 contains introduction part. Chapter 2 describes introductory information about RSVP protocol. Chapter 3 gives background information about quality of service and Chapter 4 and 5 contain detailed background of RSVP system and RSVP messages respectively. They are followed by Chapter 6. It summarizes the drawbacks of RSVP. Chapter 7 details in OPNET and its use with RSVP. Chapter 8 includes obtained simulation results and finally it is concluded with summary and conclusions.

CHAPTER 2

OVERVIEW

RSVP (Resource Reservation Protocol) is a resource reservation setup protocol for the Internet. The RSVP protocol is used by hosts to obtain specific qualities of service from the network for particular application data streams or flows. It is also used by routers to deliver quality-of-service (QoS) requests to all nodes along the path of the flows and to establish and maintain state to provide the requested service [1]. RSVP carries the request through the network, visiting each node the network uses to carry the stream. At each node, RSVP attempts to make a resource reservation for the stream. Some applications require reliable delivery of data but do not impose any stringent requirements for the timeliness of delivery. But applications such as video-conferencing, IP telephony, NetRadio require almost exact opposite: Data delivery must be timely but not necessarily reliable. Thus, RSVP was intended to provide IP networks with the capability to support the divergent performance requirements of differing application types. Originally RSVP was conceived by researchers at the University of Southern California (USC) Information Sciences Institute (ISI) and Xerox's Palo Alto Research Center (PARC). Later the Internet Engineering Task Force (IETF) specified an open version of RSVP in its RFC 2205 based basically on the USC and PARC version. 2.1 shows the position of RSVP in the TCP/IP stack.

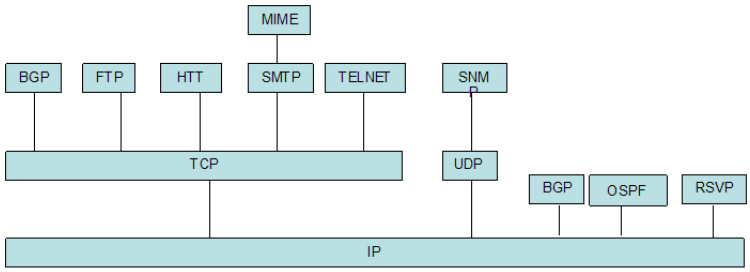


Figure 2.1: RSVP in TCP/IP Stack

Before mentioning this protocol we need to know a concept very closed to RSVP, and this is the quality of service (QoS).

CHAPTER 3

QUALITY OF SERVICE

Nowadays there are a lot of different kind of applications in Internet and each one need different properties to run. For example, distributed multimedia applications need to communicate in real-time and are sensitive to the quality of service they receive from the network. These application requirements of multimedia systems have to be transferred to the communication services and operating system. It is difficult to give a formal definition of QoS but here we can read different definitions from different organizations [2]:

- The International Telecomunication Union (ITU) refers to QoS as" A set of quality requirements on the collective behavior of one or more objects".

- The ATM Lexicon defines QoS as "A term which refers to the set of ATM performance parameters that characterize the traffic over a given virtual connection."

- The Internet Engineering Task Force (IETF) addresses QoS issues for ATM as "The demand for networked real time services grows, so does the need for shared networks to provide deterministic delivery services. Such deterministic delivery services demand that both the source application and the network infrastruc-

ture have capabilities to request, setup, and enforce the delivery of the data. Collectively these services are referred to as bandwidth reservation and Quality of Service (QoS)"

3.1 QoS Requirements

A packet stream from a source to a destination is called flow. In a connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes. The needs of each flow can be characterized by four primary parameters: reliability, delay, jitter, and bandwidth. Together these determine the QoS (Quality of Service) the flow requires. Several common applications and the stringency of their requirements are listed in Table 3.1.

Table 3.1: The Stringent of The Quality of Service Requirements

Application	**Reliability**	**Delay**	**Jitter**	**Bandwidth**
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

The first four applications have stringent requirements on reliability. No bits may be

delivered incorrectly. This goal is usually achieved by doing checksum each packet and verifying the checksum at the destination. If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually. This strategy gives high reliability. The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.

File transfer applications, including e-mail and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done. Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing have strict delay requirements. If all the words in a telephone call are each delayed by exactly 2.000 seconds, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

The first three applications are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat sensitive to that, since characters on the screen will appear in little bursts if the connection suffers much jitter. Video and especially audio are extremely sensitive to jitter. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible. For audio, a jitter of even a few milliseconds is clearly audible.

Finally, the applications differ in their bandwidth needs, with e-mail and remote login not needing much, but video in all forms needing a great deal.

3.2 Techniques for Achieving Good Quality of Service

There is no single technique provides efficient, dependable QoS in an optimum way. Instead a variety of techniques have been developed, with practical solutions often combining multiple techniques. Some of the techniques used to achieve QoS are:

Over provisioning: An easy solution is to provide so much router capacity, buffer space, and bandwidth that the packets just fly through easily. The trouble with this solution is that it is expensive. To some extent, the telephone system is over provisioned. It is rare to pick up a telephone and not get a dial tone instantly. There is simply so much capacity available there that demand can always be met.

Buffering: Flows can be buffered on the receiving side before being delivered. Buffering them does not affect the reliability or bandwidth, and increases the delay, but it smoothes out the jitter. For audio and video on demand, jitter is the main problem, so this technique helps a lot.

There is a stream of packets being delivered with substantial jitter. Packet 1 is sent from the server at t=0 sec and arrives at t=1 sec. Packets 2 undergoes more delay and takes 2 sec to arrive. As the packets arrive, they are buffered on the client machine.

At t=10 sec, playback begins. At this time, packets 1 through 6 have been buffered so that they can be removed from the buffer at uniform intervals for smooth play. Unfortunately, Packet 8 has been delayed so much that it is not available when its play slot comes up, so playback must stop until it arrives, creating an annoying gap in the music or movie. This problem can be alleviated by delaying the starting time

even more, although doing so also requires a larger buffer. Commercial web sites that contains streaming audio or video all use players that buffer for about 10 seconds before starting to play.

Traffic Shaping*: Buffering* is not always possible, for example, with videoconferencing. However, if something could be done to make the server (and hosts in general) transmit at a uniform rate, quality of service would be better. 30 packets are sent to the server machine in 30 seconds. But how do the packets go? The answer is traffic shaping which is the packets distribution in the time. Traffic shaping is a technique which smoothes out the traffic on the server side, rather than on the client side. Traffic shaping reduces congestion and thus helps the carrier live up to its promise. Such agreements are not so important for file transfers but are of great importance for real-time data, such as audio and video connections which have stringent quality-of-service requirements.

In effect, with traffic shaping the customer says to the carrier: My transmission pattern will look like this; can you handle it? If the carrier agrees, the issue arises of how the carrier can tell if the customer is following the agreement and what to do if the customer is not. Monitoring a traffic flow is called traffic policing.

The Leaky Bucket Algorithm:Imagine a bucket with a small hole in the bottom. No matter the rate at which water enters the bucket, the outflow is at a constant rate, ρ, when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full, any additional water entering it spills over the sides and is lost.

The same idea can be applied to packets. Conceptually, each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more processes within the host try to send a packet when the maximum number is already queued, the new packet is unceremoniously discarded. It is called the leaky bucket algorithm. Implementing the original leaky bucket algorithm is easy. The leaky bucket consists of a finite queue. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded. At every clock tick, one packet is transmitted.

The byte-counting leaky bucket is implemented almost the same way. At each clock tick, a counter is initialized to n. If the first packet on the queue has fewer bytes than the counter value of the counter, it is transmitted, and the counter is decremented by that number of bytes. Additional packets may also be sent, as long as the counter is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.

Token Bucket Algorithm:The leaky bucket algorithm enforces a rigid output pattern at the average rate, no matter how bursty the traffic is. For many applications, it is better to allow the output to speed up somewhat when large bursts arrive, so a more flexible algorithm is needed, preferably one that never loses data. One such algorithm is the token bucket algorithm. In this algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec.

The implementation of the basic token bucket algorithm is just a variable that counts tokens. The counter is incremented by one every ΔT and decremented by one whenever a packet is sent.

Essentially what the token bucket does is allow bursts, but up to a regular maximum length. Calculating the length of the maximum rate burst is slightly tricky. If we call the burst length S sec, the token bucket capacity C bytes, the token arrival rate ρ bytes/sec, and the maximum output rate M bytes/sec, we see that an output burst contains a maximum of C+ ρS bytes. Hence we have C+ ρ.S =MS. We can solve this equation to S=C/(M- ρ).In conclusion, S sec burst length can be accepted without losing data.

Resource Reservation:Being able to regulate the shape of the offered traffic is a good start to guaranteeing the quality of service. However, effectively using this information implicitly means requiring all the packets of a flow to follow the same route. Spraying them over routers at random makes it hard to guarantee anything. As a consequence, something similar to a virtual circuit has to be set up from the source to the destination, and all the packets that belong to the flow must follow this route. Once we have a specific route for a flow, it becomes possible to reserve resources along that route to make sure the needed capacity is available[3].

There are three basic levels of end-to-end QoS that can be provided across a heterogeneous network [2, 4, 5]:

Best-effort service: Internet offers a service based on the" best effort" delivery model. This model delivers data to the destination as soon as possible, but with

no commitment as to bandwidth or latency. For this reason,"the best effort" model is inadequate for applications requiring timeless. The FIFO queues are the best characterization of this service.

Differentiated service (DiffServ): Is traffic that requires timeliness of delivery and that varies its rate accordingly. It means that some traffic is treated better than the rest, but not a fast and guarantee. For instance, MPEG-II video averages about 3 to 7 Mbps, depending on the amount of change in a picture. As an example, 3 Mbps might be a picture of a painted wall, although 7 Mbps would be required for a picture of waves on the ocean. This service is as well referred as Controlled delay service.

Integrated service (IntServ): This service requires a guaranteed transmission rate from its source to its destination. The objective of IntServ is to have only one IP network which has best-effort service and flows with a QoS. This is provided through QoS tools RSVP and CBWFQ. It is called as well guaranteed bit-rate service. It shows different treating of data flow in RSVP and Best Effort.

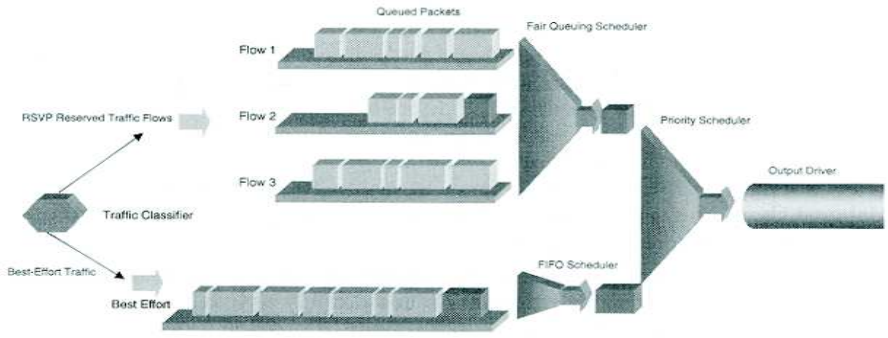


Figure 3.1: Service Types

The reader can find more information about the Controlled-Load and Guaranteed Quality of Service in the next references [6, 7].

3.3 Real Time Protocols

The Integrated Services working group in IETF (Internet Engineering Task Force) developed an enhanced Internet service model called Integrated Services that includes best-effort service and real-time service, see RFC 1633. Integrated Services allows applications to configure and manage a single infrastructure for multimedia applications and traditional applications. The real-time service will enable IP networks to provide quality of service to multimedia applications. Resource Reservation Protocol (RSVP) , together with Real-Time Transport Protocol(RTP) , Real-Time Control Protocol(RTCP), Real-Time Streaming Protocol(RTSP),provides a working foundation for real-time services. The most widely used transport-level protocol is TCP. Although TCP has proven its value in supporting a wide range of distributed applications, it is not suited for use with real-time distributed applications. By real-time distributed applications, we mean one in which a source is generating a stream of data at a constant rate, and one or more destinations must deliver that data to an application at the same constant rate. Examples of such applications include audio, video conferencing, live video distribution, shared workspaces, remote medical diagnosis, telephony, command and control systems, distributed interactive simulations, games, and real-time monitoring. A number of features of TCP disqualify it for use as transport protocol for such applications:

1. TCP is a point-to-point protocol that sets up connection between two end points. Therefore, it is not suitable for multicast distributions.

2. TCP includes mechanisms for retransmission of lost segments, which then arrive

out of order. Such segments are not usable in most real-time applications.

3. TCP contains no convenient mechanism for associating timing information with segments, which is another real-time requirement.

The other widely used transport protocol, UDP, does not exhibit the first two characteristics listed but, like TCP, does not provide timing information. By itself UDP does not provide any general purpose tools useful for real-time applications [8].

Real time protocols cover specific needs by applications with real-time characteristics. Real-time applications have specific requirements from the lower layers, mainly in terms of packet loss, delay, and jitter. Traditional transport protocols such as TCP and UDP have been designed for general use and are not specialized for such specific purposes. In particular, real-time protocols have to be able to deliver high throughput, handle multicast, manage the transmission quality and be friendly to the rest of the traffic and more importantly to the congestion-sensitive TCP traffic [9].

The most widely used protocols for carrying real-time application data are RTP and RTCP protocols.

3.3.1 Real Time Protocol(RTP)

Real-time transport protocol (RTP) is an IP-based protocol providing support for The transport of real-time data such as video and audio streams. The services provided by RTP include time reconstruction, loss detection, and content identification. RTP is primarily designed for multicast of real-time data, but it can be also used in unicast.

It can be used for one-way transport such as video-on-demand as well as interactive services such as internet telephony [10]. RTP is designed to work in conjunction with the auxiliary control protocol RTCP to get feedback on quality of data transmission and information about participants in the on-going session. Packets sent on the internet have unpredictable delay and jitter. But multimedia applications are require appropriate timing in data transmission and playing back.

RTP provides time stamping, sequence numbering and other mechanisms to take care of the timing issues. Through these mechanisms, RTP provides end-to-end transport for real-time data over datagram networks.

RTP is typically run on top of UDP to make use of its multiplexing and checksum functions. UDP was chosen as the target transport protocol for RTP because of two reasons:

First, RTP is primarily designed for multicast; the connection-oriented TCP does not scale well and therefore is not suitable.

Second, for real-time data, reliability is not as important as timely delivery. Even more, reliable transmission provided by retransmission as in TCP is not desirable. RTP and RTCP packets are usually transmitted using UDP/IP service.

3.3.2 RTP Control Protocol (RTCP)

RTCP is the control protocol that works in conjunction with RTP. It provides support for real-time conferencing for large groups within an internet, including source

identification and support for gateways and multicast-to-unicast translators. In an RTP session, participants periodically send RTCP packets to convey feedback on quality of data delivery and information of membership. It is standardized in RFC 1889 and RFC 1890. RTCP provides QoS monitoring and congestion control, source identification, inter-media synchronization, and control information scaling services.

3.3.3 Real Time Streaming Protocol(RTSP)

Instead of storing large multimedia files and playing back, multimedia data is usually sent across the network in streams. Streaming breaks data into packets with size suitable for transmission between the servers and clients. The real-time data flows through the transmission, decompress the second and playing back pipeline just like a water stream. A client can play first packet, decompress the second, while receiving the third. Thus the user can start enjoying the multimedia without waiting to the end of transmission.

RTSP, the Real Time Streaming Protocol, is a client-server multimedia presentation protocol to enable controlled delivery of streamed multimedia data over IP network. It provides "VCR style" remote control functionality for audio and video streams, like pause, fast forward, reverse, and absolute positioning. Source of data include both live data feeds and stored clips.

RTSP is an application-level protocol designed to work with lower-layer protocols like RTP, RSVP to provide complete streaming service over internet. It provides means for choosing delivery channels (such as UDP, multicast UDP and TCP) and delivery

mechanisms based upon RTP.

RTSP aims to provide the same services on streamed audio and video just HTTP does for text and graphics. It is designed intentionally to have similar syntax and operations so that most extension mechanisms to HTTP can be added to RTSP.

As many real-time applications have been developed in the internet, the best effort delivery model became inadequate for these new applications. TCP, used widely in current internet, is not well suited to real-time applications. Instead, Real-Time Transport Protocol (RTP) is usually implemented on top UDP, which is better adapted to real-time applications. However, this protocol mechanism is not enough to guarantee a specific quality of service (QoS) for a session between a sender and a receiver [11].

CHAPTER 4

DETAILED BACKGROUND OF RSVP SYSTEM

The RSVP protocol performs a reservation for each flow requiring QoS services; a flow is defined by five tuples (source IP address, destination IP address, transport protocol, source port, and destination port). Each flow needs several RSVP messages, to request, maintain and release the required resources.

With an RSVP based quality of service architecture there are two basic elements: sources and destinations, all of them run RSVP daemons that participate in RSVP protocol and exchange RSVP messages on behalf of their hosts. They exchange basically two types of messages: PATH and RESV. The RSVP source sends a PATH message which is encapsulated in IP or UDP datagrams. The message travels through the network to the destination.

When it is received by the destination, if it wants to make a reservation for the particular RSVP flow, it responds with a RESV message and it traverses the reverse path back to the sender. Otherwise, a RESV ERROR message is issued and is sent back to the receiver. An end-to-end reservation is successfully established when the RESV message reaches the sender and is successfully processed by the RSVP daemon on the sender and in all the other nodes in the middle.

A multicast reservation session can also be made. In this case the sender sends the PATH messages to a multicast group address. As in the case of unicast, the path messages travels through the network to all the members of the multicast group. When PATH messages reach the receivers, each receiver independently decides if it wants to request a reservation for the session. Each receiver can potentially request for different reservations for the same session [12]. Figure 4.1 shows an example, where S1 and S2 are sources, and D1, D2 and D3 are destinations of data. D1, D2 and D3 are members of the same multicast group and S1 and S2 send messages to this multicast group. But we can see in the figure that not all the destinations make the same reservation. D3 sends RESV messages to S1 and S2, so it accepts all the reservations. D2 only accepts one of the reservation requests sending one RESV message to S1. But D1 does not want to make any reservation and does not send any RESV message to S1 and S2.

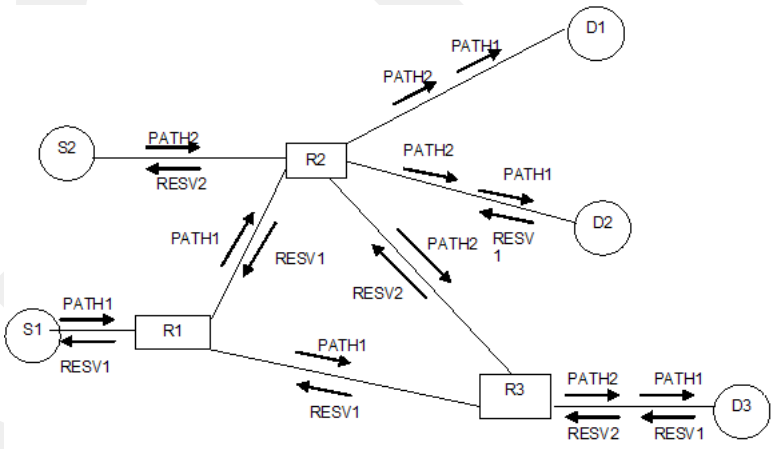


Figure 4.1: PATH and RESV Messages Flows in RSVP

It has to be clear that RSVP is a protocol to negotiate a quality of service for an specific application and it is not a routing protocol. It uses the routing table in routers to determine routes to the appropriate destinations. So it was designed to

interoperate with existing unicast and multicast IP routing protocols [13].

4.1 Data Flows

A session in RSVP is defined to be a data flow with a particular destination and transport-layer protocol [1]. An RSVP session is defined by 3-tuple consisting on:

- Destination address: Destination IP address of the packages (unicast or multicast) IPv4/UDP SESSION objects: $Class = 1$, $C\text{-}Typ = 1$, IPv4 DestAddress (4 bytes) IPv6/UDP SESSION objects: $Class = 1$, $C\text{-}Type = 2$, IPv6 DestAddress (16 bytes)
- Protocol ID: Protocol ID is the identifier ID of the IP protocol
- Destination port (optional): UDP/TCP destination port field

The optional Destination Port parameter is a "generalized destination port", i.e., some further demultiplexing point in the transport or application protocol layer. Note that it is not strictly necessary to include Destination Port in the session definition when Destination Address is multicast, since different sessions can always have different multicast addresses. However, Destination Port is necessary to allow more than one unicast session addressed to the same receiver host.

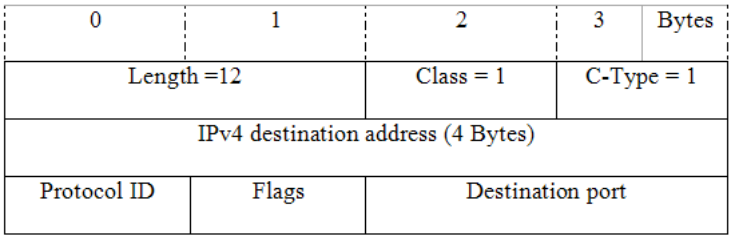


Figure 4.2: IPv4 UDP Session Object.

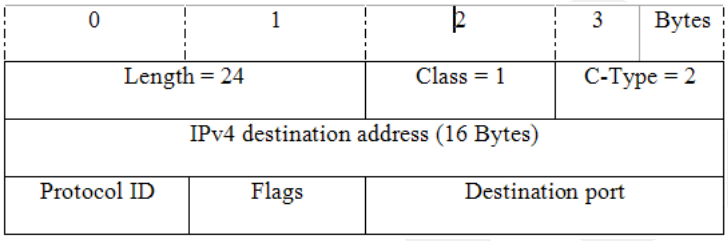


Figure 4.3: IPv6 UDP Session Object.

Figure 4.2 and Figure 4.3 show the format of UDP session objects for IPv4 and IPv6 addresses with $C\text{-}Type = 1$ and $C\text{-}Type = 2$, respectively. It can be extended to accommodate other address by defining a new C-Type. The format contains fields such as the destination IP address, the IP protocol identifier, and the destination TCP or UDP port of a data flow. There are two types of these objects to support IPv4 and IPv6 addressing. These addresses can be either unicast or multicast. A flag field can be set if a host is not capable of policing and wants the edge network device to perform policing. All RSVP messages require this object to identify a flow.

4.2 Traffic control

Quality of service is implemented for a particular data flow by mechanisms collectively called "traffic control". These mechanisms include (1) a packet classifier, (2) admission control, and (3) a "packet scheduler" or some other link-layer-dependent mechanism to

determine when particular packets are forwarded. The "packet classifier" determines the QoS class (and perhaps the route) for each packet. For each outgoing interface, the "packet scheduler" or other link-layer-dependent mechanism achieves the promised QoS. Traffic control implements QoS service models defined by the Integrated Services Working Group.

During reservation setup, an RSVP QoS request is passed to two local decision modules, "admission control" and "policy control". Admission control determines whether the node has sufficient available resources to supply the requested QoS. Policy control determines whether the user has administrative permission to make the reservation. If both checks succeed, parameters are set in the packet classifier and in the link layer interface (e.g., in the packet scheduler) to obtain the desired QoS. If either check fails, the RSVP program returns an error notification to the application process that originated the request. Figure 4.4 shows the main modules in a host and a router.

RSVP protocol mechanisms provide a general facility for creating and maintaining distributed reservation state across a mesh of multicast or unicast delivery paths. RSVP itself transfers and manipulates QoS and policy control parameters as opaque data, passing them to the appropriate traffic control and policy control modules for interpretation.

Since the membership of a large multicast group and the resulting multicast tree topology are likely to change with time, the RSVP design assumes that state for RSVP and traffic control state is to be built and destroyed incrementally in routers and hosts. For this purpose, RSVP establishes "soft" state; that is, RSVP sends

periodic refresh messages to maintain the state along the reserved path(s). In the absence of refresh messages, the state automatically times out and is deleted [1].

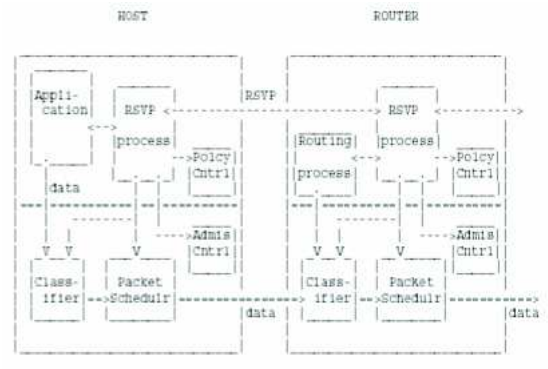


Figure 4.4: RSVP in Hosts and Routers.

4.3 Reservation styles

Reservation style indicates to the network element that an aggregation of reservation request is possible for a multicast group. Resource reservation controls how much bandwidth is reserved, whereas reservation filter determines the packets that can make use of this reservation. RSVP supports three styles of reservation. A description of these styles is provided in the following subsections. If we have different senders for the same RSVP session, then we have two modes:

- Distinct Reservation: creates a different reservation for each upstream sender.
- Shared Reservation: creates a shared reservation for specified senders.

But we have another option that controls the set of senders. With this option there are also two options:

- **Explicit:** select a list of the senders.

- **Wildcard:** selects all the senders for the session. And now if we mix these modes, then as we can see in Figure 4.5, there exist the Fixed-Filter style (FF), the Shared-Explicit style (SE), and the Wildcard-Filter (WF):

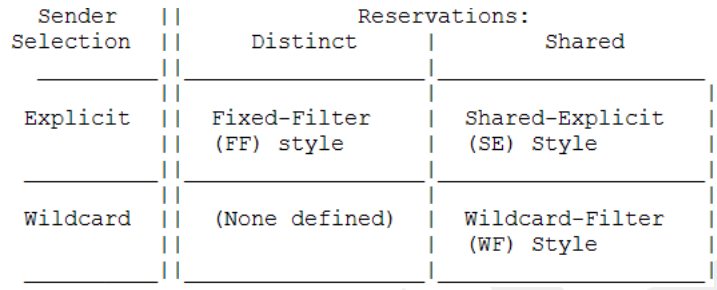


Figure 4.5: Reservation Attributes and Styles

The Wildcard-Filter (WF) style creates a single reservation shared by all flows from all upstream senders. The Shared-Explicit style (SE) creates a single reservation shared by selected upstream senders, so is the same than the Wildcard-Filter but with not all the senders. And the last style is the Fixed-Filter, which creates a distinct reservation for data packets from a particular sender. This is the last style because there is no defined style for a distinct reservation in a Wildcard sender selection. WF and SE are appropriate for multicast applications in which multiple data sources are unlikely to transmit simultaneously.

4.4 Examples of Reservation Styles

4.4.1 Wildcard Filter

The wildcard filter (and shared explicit) style reservation is suitable for multicast sessions where sources are not likely to send information at the same time. Typically, audio applications are suitable for this style since only a limited number of participants

can converse with each other simultaneously. A reservation slightly exceeding the requirements for a single speaker (for over speaking and interjections) will be sufficient for this style.

We look at the wildcard-filter style using an example in Figure 4.6.These examples use a rate of Kbps for simplification (in reality token bucket parameters are used).The example uses a multicast session with three senders S1,S2 and S3 and three receivers H1,H2,H3.The senders S1 and S2 as well as receivers H1 and H2 are shown to be on LAN segment capable of implementing traffic priority schemes. Following are the requirements of receivers:

-H1 wants to reserve 3 Kbps

-H2 wants to reserve 2 Kbps

-H3 wants to reserve 4 Kbps

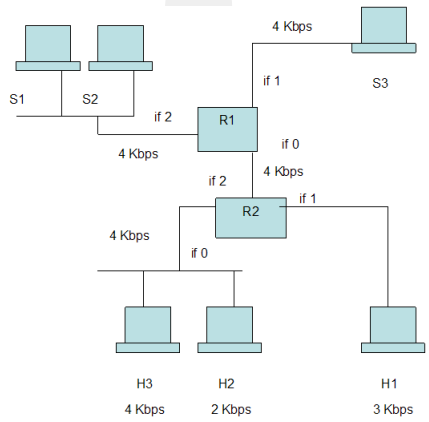


Figure 4.6: Example of Wildcard Filter

Reservation for H2 and H3 are merged 4 Kbps (if0, R2).

Another request comes from H1 on if1 at R2 for 3 Kbps.R2 sends a merged request via (if2, R2) of 4 Kbps.

It is larger of (if1, R2) 3 Kbps and (if0, R2) 4Kbps.Router R1 forwards a 4 Kbps request on if1 and if2 (to S1, S2 and S3).

An important point to note here is that the source is not identified and that merger of requests at routers does not use the sum of the incoming requests, but takes the larger of the two values.

4.4.2 Shared Explicit

The shared-explicit-filter style reservation is similar to wildcard-filter, with the only difference that here, sender are identified. The reservation is shared among all senders in the list. Figure 4.7 shows an example of the shared-explicit-filter style reservation. In this case, following are the requirements of receivers:

-H1 wants to reserve 1 Kbps for S1 and S2.

-H2 wants to reserve 3 Kbps for S1 and S3.

-H3 wants to reserve 2 Kbps for S2.

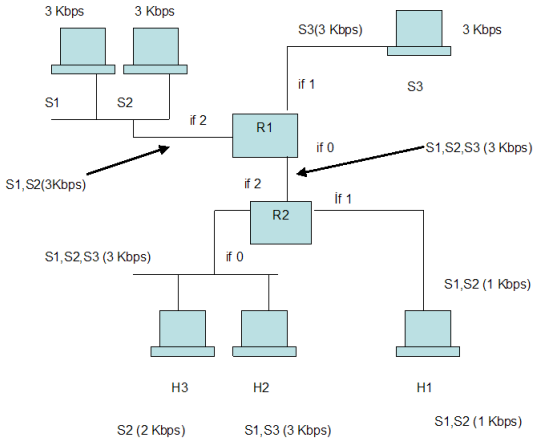


Figure 4.7: Example of Wildcard Filter

A reservation for sources S1, S2 and S3 from H2 and H3 on (if0, R2) is merged to 3 Kbps. Another request comes from H1 on if1 of R2 for 1 Kbps to S1 and S2.The requests on if0 and if1 of router R2 are merged and forwarded on if2 as 3 Kbps for S1, S2, and S3.The requests received on if0 of router R1 are forwarded as follows:

- On if2 3 Kbps for S1 and S2.
- On if1 3 Kbps for S3.

4.4.3 Fixed Filter

The fixed-filter style reservation is suitable for applications such as videoconferencing, where one window is required for each sender and all these windows need to be updated simultaneously. Fixed-style reservation requires that receivers identify the source from which they want to receive the reservation along with the bandwidth required. Bandwidth is not shared (between sources), since reservations are made for a particular source.

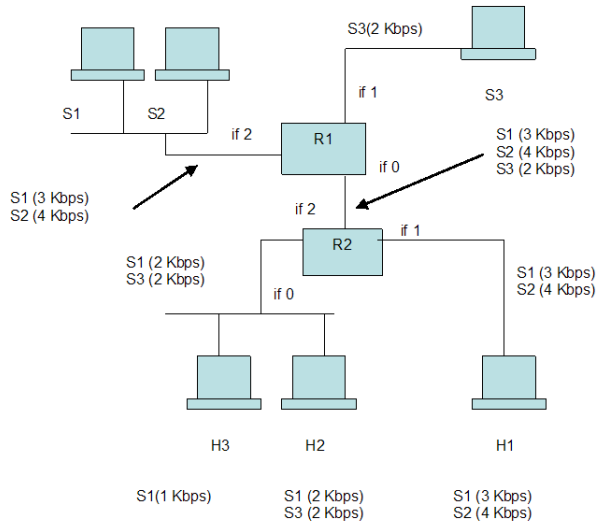


Figure 4.8: Example of Wildcard Filter

Figure 4.8 shows how the fixed-filter style reservation can be used. Following are the requirements of receivers:

- H1 wants to reserve3 Kbps for S1 and 4 Kbps for S2.
- H2 wants to reserve 2 Kbps for S1 and 2 Kbps for S3.
- H3 wants to reserve 1 Kbps for S1.

The reservation for source S1 from H2 and H3 is merged to 2 Kbps at (if0, R2).The reservation for source S3 of 2 Kbps from H2 arrives at (if0, R2).Another request comes from H1 on if1 of R2 for 3 Kbps to S1 and 4Kbps to S2.The requests on if0 and if1 of router R2 are merged and forwarded on if2 as follows:

- 3 Kbps for S1;
- 4 Kbps for S2;
- 2 Kbps for S3.

The requests received on if0 of router R1 are forwarded as follows:

- On if2 3 Kbps for S1 and 4 Kbps for S2;
- On if1 2 Kbps for S3.

CHAPTER 5

RSVP MESSAGES

As we said before, an RSVP sender transmits PATH messages downstream and store's information in each node along the way. This information includes the IP address of each previous hop in the traffic path which will be used to forward the subsequent RESV message. The RSVP receiver sends the RESV message upstream to the sender, which creates a reservation state in each node along the traffic path, following the same way than the previous PATH message. An RSVP message consists on the heading and the object. The heading is very important because in it there is the type of message and with it each node can recognize if the message is a PATH, RESV or an Error message. Then with the object the sender or the receiver specifies the reservation style and the quality of service. In the next figures we can see the heading format and the object format. The RSVP objects travel in RSVP messages and contain specific information for different purposes. We will see it in more detail in next section [13].

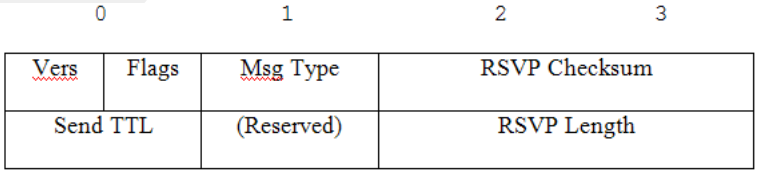


Figure 5.1: RSVP Header

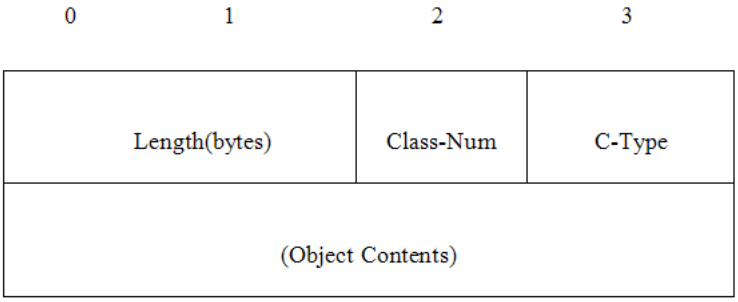


Figure 5.2: RSVP Object

5.1 PATH Messages

This message as we told before is sent by the host which wants to make a QoS reservation. This message travels along the network using the way given by the routing protocols and they keep the 'path state' which will be followed by the RESV messages to ind the sender. It contains the IP address of the previous node, to allow the RESV messages to ind the same way back to the sender. This message includes the Sender Template, the Sender TSpec and the Adspec.

Sender Template: describes the format of the data traffic that the sender will send. The Sender Template contains a parameter which identifies the sender's flow from other flows that share the same RSVP session on the same link; this parameter is called Filter Spec (Filter specification).

- Sender TSpec: describes the traffic flow that the sender wants to generate. This parameter is not modified by the intermediate receivers [14].
- Adspec: collects information from the intermediate network elements.

When a node receives a PATH message, it modifies the Adspec with resource infor-

mation and sends it to the next downstream hop. This parameter is very important for receivers because it allow them to choose a QoS control service and determine the right reservation parameters. While the PATH message goes toward the receiver, the nodes in the middle include some information in the Adspec object as delay and bandwidth estimates and various parameters used by specific QoS control services. The information in the Adspec is updated each time the PATH message passes through a node. At each node the Adspec is processed by the traffic-control module. Then it updates the Adspec identifying the services specified in the Adspec and calling each process to update it. If there is any QoS specified in the Adspec that the node doesn't support, then a flag is set to report this to the receiver.Figure 5.3 shows perfectly how is established the path state between the RSVP sender and receiver.

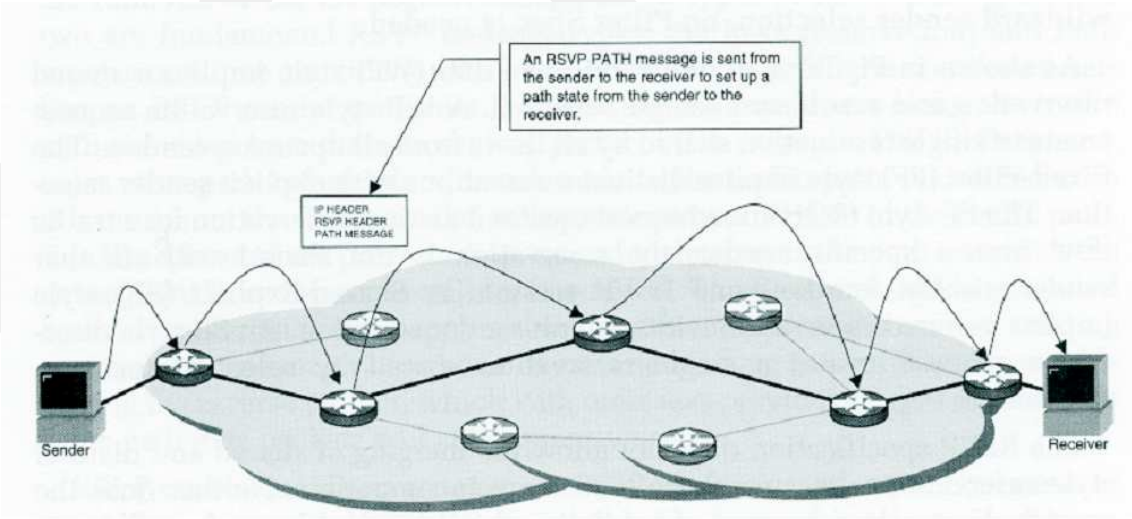


Figure 5.3: Path State

5.2 RESV Messages

Each receiver sends reservation messages (RESV) through the network to the senders. They follow the same way as the PATH messages but it the opposite direction. They create and keep a 'reservation state' for each node in the way as we can see in Figure 5.4.

5.2.1 Reservation Model

A RESV message contains information about the reservation style, the Flowspec object and the Filterspec that identify the sender(s). This pair is called a flow descriptor.

- Flowspec: specifies the desired QoS and is used to set the parameters in the packet scheduler of each node.
- Filter spec: defines the set of data packets (defined in terms of sender parameters) to receive the QoS defined by the flowspec and is used to set parameters by the packet classifier. The format differs between the IP versions.

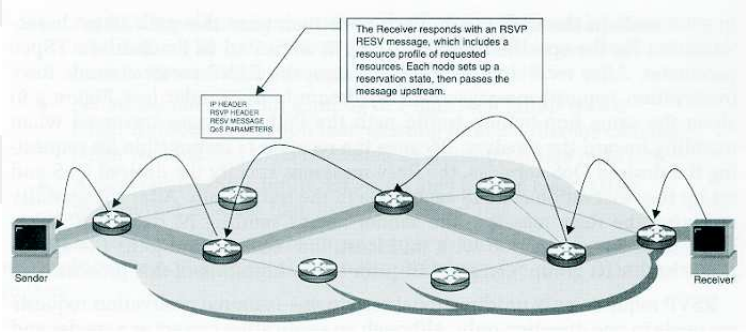


Figure 5.4: Reservation State

To get the appropriate QoS between a sender and a receiver is necessary to communicate some information to the nodes between them. Some of this information is the Receiver Tspec and the Receiver Rspec. The first one describes the nvoke the desired QoS. This information is contained in the Flowspec and carried in the RESV messages. The information in the Flowspec may be modified at any intermediate node in the traffic path. In each node an RSVP reservation request take two actions:

Make a reservation on the link: The RSVP process passes the request to the admission and policy control modules. If there is some fail in some of them then the process sends an error message to the receiver(s). But if it passes both tests then the packet classifier is setup and it select that data packets defined by the Filterspec and interact with the link-layer to find the QoS defined by the Flowspec.

Forward the request upstream: The set of senders which a reservation request is propagated is called SCOPE. We should differentiate between a reservation re-

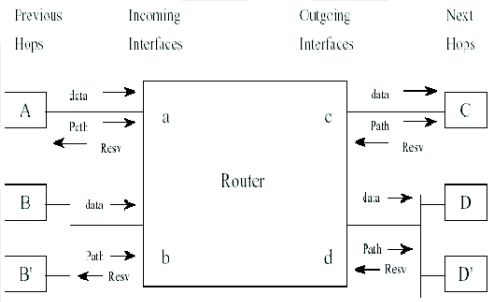


Figure 5.5: Router Using RSVP

quest when a node forwards it and when a node receive it. Because the traffic control mechanism modifies the flowspec hop-by-hop and reservations from different downstream branches of the multicast tree from the same sender (or set of senders) must be merged as reservations travel upstream [1]. Figure 5.5 shows the incoming and

outgoing interfaces of a router.

5.2.2 Soft State

The soft state is the maintenance of the reservation state along a particular traffic path, which is maintained by RESV messages. This mechanism is necessary because Internet is a dynamic network, it means that some routers can stop working or others can appear.

The soft state has to be maintained in each node to allow changes in the network without ask to the end nodes. To keep the soft state is necessary to refresh each node after some specific time. When a path changes, the next PATH message will start the 'path state' in the new way. The next RESV message will then set up the new soft state. In one node, the soft state can be deleted after a certain period of time (Timeout) or with some specific messages called Teardown. We will tell more about these messages in the next section.

5.3 Teardown Messages

These messages are used to remove the path or the reservation state. It is not necessary to remove the last reservation, but it is recommended for all the end nodes to send a teardown message when an application finish. There are two kinds of these messages:

- Path Tear: it goes through all receivers by removing the 'path state'.

- Resv Tear: it goes through all senders removing the 'reservation state'.

  It can be generated by an application in an end-node or by a router as a result of a timeout.

5.4 Error Messages

There are two types of error messages:

- PathErr: it is generated when there is an error sending a PATH message.
- ResvErr: it is generated when there is an error sending a RESV message.

The PathErr is sent to the sender with the type of the error and the IP address of the node who has detected the error. The ResvErr is directed to the flow receiver who requested the reservation. These messages do not modify the path state in the nodes through which they pass.

5.5 Merging

The concept of merging appears in multicast traffic and RSVP because of the packet delivering at each node. As it was told before, each RESV message carries the Flowspec which specifies the desired QoS. Then if there are different RSVP receivers which each one sends their RESV message, in one of the routers in the middle of the way between the sender and the receiver, it will receive different RESV messages. And it will have to send to the same RSVP sender. So it has to merge the Flowspecs. The

largest Flowspec from all the merged Flowspecs is used to define the single merged Flowspec.

It is very important to mention that different reservation styles cannot be merged because they are incompatible. In Figure 5.6, it is shown how merging works with two receivers and one sender [13].

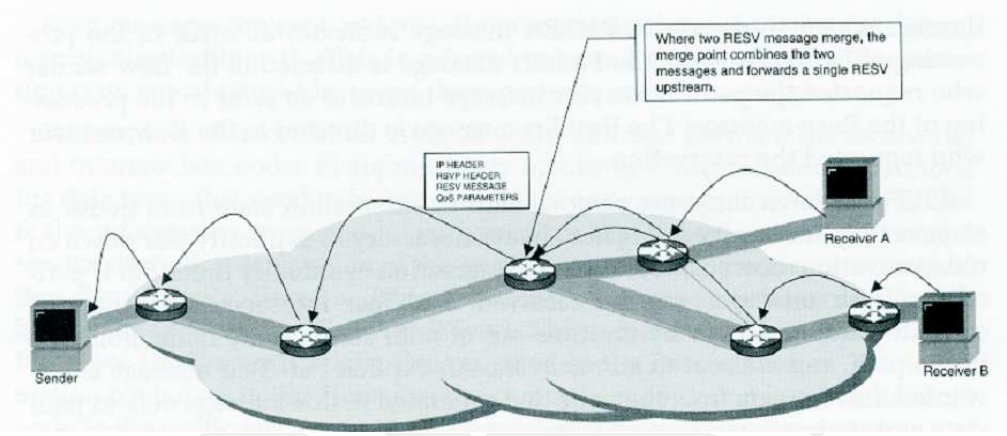


Figure 5.6: Merging

CHAPTER 6

RSVP PROTOCOL PROBLEMS

6.1 Issues Affecting Deployment of RSVP

Wide scale deployment of RSVP must be approached with care, as there remains a number of an outstanding issue that may affect the success of deployment.

6.1.1 Scalability

The resource requirements for running RSVP on a router increase proportionally with the number of separate sessions (i.e., RSVP reservations). Thus, supporting numerous small reservations on a high-bandwidth link may easily overly tax the routers and is inadvisable.

Furthermore, implementing the packet classification and scheduling capabilities currently used to provide differentiated services for reserved flows may be very difficult for some router products or on some of their high speed interfaces.

These scaling issues imply that it will generally not be appropriate to deploy RSVP

on high-bandwidth backbones at the present time. Looking forward, the operators of such backbones will probably not choose to naively implement RSVP for each separate stream.

Rather, techniques are being developed that will, at the "edge" of the backbone, aggregate together the streams that require special treatment. Within the backbone, various less costly approaches would then be used to set aside resources for the aggregate as a whole, as a way of meeting end-to-end requirements of individual flows.

6.1.2 Security Considerations

The RSVP WG submission for Proposed Standard includes two security related documents [15, 16]. The study in [15] addresses denial and hijacking or theft of service attacks. The study in [16] addresses RSVP mechanisms for data flows that themselves use IPSEC.

The first document is proposed to protect against spoofed reservation requests arriving at RSVP routers; such requests might be used to obtain service to unauthorized parties or to lock up network resources in a denial of service attack. Modified and spoofed reservation requests are detected by use of hop-by-hop MD5 checksums (in an Integrity Object) between RSVP neighbor routers.

As described, RSVP hop-by-hop authentication assumes that key management and distribution for routers is resolved and deployed. Until an effective key infrastructure is in place, manually keyed session integrity might be used. That RSVP needs

an effective key infrastructure among routers is not unique to RSVP: it is widely acknowledged that there are numerous denial of service attacks on the routing infrastructure (quite independent of RSVP) that will only be resolved by deployment of a key infrastructure. Theft of service risks will require the user to deploy with caution. An elementary precaution is to configure management logging of new and changed filter specifications in RSVP-enabled infrastructure. The second security-related document provides a mechanism for carrying flows in which the transport and user octets have been encrypted (RFC 1827). Although such encryption is highly beneficial to certain applications, it is not relevant to the functional security of RSVP or reservations.The following section on Policy Control includes additional discussion of RSVP authorization security.

6.1.3 Policy Control

Policy control addresses the issue of who can, or cannot, make reservations once a reservation protocol can be used to set up unequal services. Currently, the RSVP specification defines a mechanism for transporting policy information along with reservations. However, the specification does not define policies themselves.

At present, vendors have stated that they will use the RSVP-defined mechanism to implement proprietary policies. The RSVP WG is chartered to specify a simple standardized policy object and complete simple mechanisms for session use of the Integrity object in the near future.

This applicability statement may be updated at the completion of the WG's

charter. Before any decision to deploy RSVP, it would be wise to ensure that the policy control available from a vendor is adequate for the intended usage. In addition to the lack of documented policy mechanisms in any of the policy areas (such as access control, authorization, and accounting), the community has little experience with describing, setting and controlling policies that limit Internet service. Therefore it is likely that vendor solutions will be revised often, particularly before the IETF has developed any policy specification [17].

6.2 RSVP Protocol Performance Issues

6.2.1 Processing Overhead

By "processing overhead" we mean the amount of processing required to handle messages belonging to a reservation session. This is the processing required in addition to the processing needed for routing an (ordinary) IP packet. The processing overhead of RSVP originates from two major issues:

1. **Complexity of the protocol elements:** First, RSVP itself is per-flow based; thus the number of states is proportional to RSVP session number. Path and Resv states have to be maintained in each RSVP router for each session (and Path state also has to record the reverse route for the correspondent Resv message).

   Second, being receiver-initiated, RSVP optimizes various merging operations for multicast reservations while the Resv message is processed. To handle multicast,

other mechanisms such as reservation styles, scope object, and blockade state are also required to be presented in the basic protocol. This not only adds sources of failures and errors, but also complicates the state machine. " Third, the same RSVP signaling messages are used not only for maintaining the state, but also for dealing with recovery of signaling message loss and discovery of route change. Thus, although protocol elements that represent the actual data (e.g., QoS parameters) specification are separated from signaling elements, the processing overhead needed for all RSVP messages is not marginal. " Finally, the possible variations of the order and existence of objects increases the complexity of message parsing and internal message and state representation.

2. **Implementation-specific Overhead:** There are two ways to send and receive RSVP messages: either as "raw" IP datagrams with protocol number 46, or as encapsulated UDP datagrams, which increase the efficiency of RSVP processing. Typical RSVP implementations are user-space daemons interacting with the kernel; thus, state management, message sending, and reception would affect the efficiency of the protocol processing. In [18], It is stated that state (memory) management can use up to 17-18 % of the total execution cost, but it is possible to decrease that cost to 6-7%, if appropriate action is taken to replace the standard memory management with dedicated memory management for state information. RSVP/routing, RSVP/policy control, and RSVP/traffic control interfaces can also pose different overhead depending on implementation. For example, the RSVP/routing overhead has been measured to be approximately 11-12% of the total execution cost.

6.2.2 Bandwidth Consumption

By "bandwidth consumption" we mean the amount of bandwidth used during the lifetime of a session: to set up a reservation session, to keep the session alive and finally closing it. RSVP messages are sent either to trigger a new reservation or to refresh an existing reservation. In standard RSVP, Path/Resv messages are used for triggering and refreshing/recovering reservations, identically, which results in an increased size of refresh messages. The hop-by-hop refreshment may reduce the bandwidth consumption for RSVP, but could result in more sources of error/failure events. The study in [19] presents a way to bundle standard RSVP messages and reduces the refreshment redundancy by Srefresh message. Thus, the following formula represents the bandwidth consumption in bytes for an RSVP session lasting n seconds:

$$\mathrm{F(n)} = (\mathrm{bP} + \mathrm{bR}) + ((\mathrm{n/Ri}) * (\mathrm{bP} + \mathrm{bR})) + \mathrm{bPt} \tag{6.1}$$

Where, bP: IP payload size of Path message bR: IP payload size of Resv message bPt: IP payload size of Path Tear message Ri: refresh interval For example, for a simple Controlled Load reservation without security and identification requirements (where bP is 172 bytes, bR is 92, bPt is 44 bytes, and Ri is 30 seconds), the bandwidth consumption would be as follows:

$$\mathrm{F(n)} = (172 + 92) + ((\mathrm{n}/30) * (172 + 92)) + 44 \tag{6.2}$$

$$= 308 + (264n/30)bytes$$

RSVP recommended using in small local network because of scalability problem. However RSVP still have some problems in small network such as signaling traffic overhead, limitation of the resource reservation [20].

6.3 Routing Reservations

There is a fundamental interaction between resource reservation set up and routing, since reservation requires the installation of flow state along the route of data packets. If and when a route changes, there must be some mechanism to set up a reservation along the new route. RSVP protocol works under the assumption that RSVP software must exists in the receivers, senders and routers. There are four routing issues faced by a reservation setup protocol such as RSVP.

1. Find a route that supports resource reservation. This is simply "type-of-service" routing, a facility that is already available in some modern routing protocols.

2. Find a route that has sufficient unreserved capacity for a new flow. Early experiments on the ARPANET showed that it is difficult to do load-dependent dynamic routing on a packet-by-packet basis without instability problems. However, instability should not be a problem if load-dependent routing is performed only at reservation setup time. Two different approaches might be taken to finding a route with enough capacity. One could modify the routing protocol(s) and interface them to the traffic control mechanism, so the route computation can consider the average recent load. Alternatively, the routing protocol could be (re-)designed to provide multiple alternative routes, and reservation setup could be attempted along each in turn.

3. Adapt to a route failure When some node or link fails, adaptive routing finds an alternate path. The periodic refresh messages of RSVP will automatically request a reservation along the new path. Of course, this reservation may fail be-

cause there is insufficient available capacity on the new path. This is a problem of provisioning and network engineering, which cannot be solved by the routing or setup protocols. There is a problem of timeliness of establishing reservation state on the new path. The end-to-end robustness mechanism of refreshes is limited in frequency by overhead, which may cause a gap in real-time service when an old route breaks and a new one is chosen.

4. Adapt to a route change (without failure) Route changes may occur even without failure in the affected path. Although RSVP could use the same repair techniques as those described in (3), this case raises a problem with the robustness of the QoS guarantees. If it should happen that admission control fails on the new route, the user will see service degradation unnecessarily and capriciously, since the original route is still functional. To avoid this problem, a mechanism called "route pinning" has been suggested. This would modify the routing protocol implementation and the interface to the classifier, so that routes associated with resource reservations would be "pinned". The routing protocol would not change a pinned route if it was still viable. It may eventually be possible to fold together the routing and reservation setup problems, but we do not yet understand enough to do that. Furthermore, the reservation protocol needs to coexist with a number of different routing protocols in use in the Internet. Therefore, RSVP is currently designed to work with any current-generation routing protocol without modification. This is a short-term compromise, which may result in an occasional failure to create the best, or even any, real-time session, or occasional service degradation due to a route change. We expect that future generations of routing protocols will remove this compro-

mise, by including hooks and mechanisms that, in conjunction with RSVP, will solve the problems (1) through (4) just listed. They will support route pinning, notification of RSVP to trigger local repair, and selection of routes with "IS" support and adequate capacity [21].

6.4 QoS-Based Routing and Resource Reservation

It is important to understand the difference between QoS-based routing and resource reservation [22]. While resource reservation protocols such as RSVP [23]. provide a method for requesting and reserving network resources, they do not provide a mechanism for determining a network path that has adequate resources to accommodate the requested QoS. Conversely, QoS-based routing allows the determination of a path that has a good chance of accommodating the requested QoS, but it does not include a mechanism to reserve the required resources.

Consequently, QoS-based routing is usually used in conjunction with some form of resource reservation or resource allocation mechanism [22]. Simple forms of QoS-based routing have been used in the past for Type of Service (TOS) routing [24]. In the case of OSPF, a different shortest-path tree can be computed for each of the 8 TOS values in the IP header [25]. Such mechanisms can be used to select specially provisioned paths but do not completely assure that resources are not overbooked along the path. As long as strict resource management and control are not needed, mechanisms such as TOS-based routing are useful for separating whole classes of traffic over multiple routes. Such mechanisms might work well with the emerging

Differential Services efforts [26].

Combining a resource reservation protocol with QoS-based routing allows fine control over the route and resources at the cost of additional state and setup time. For example, a protocol such as RSVP may be used to trigger QoS-based routing calculations to meet the needs of a specific flow.

CHAPTER 7

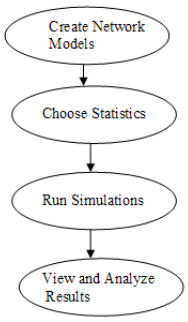
OPNET

7.1 Overview of OPNET

Network simulations are created for the purpose of studying a system's performance and behavior under various circumstances. In broad terms, the process of building and simulating a network model is a repetitive process that involves four steps, which are graphically illustrated in Figure 7.1. The first and non-repetitive step involves the initial specification and development of a system model. Since the objective of most system modeling is to measure a system's performance and to make observations regarding its behavior, data collection must be specified and simulations must be executed (for sufficient time and under variable behavior), which is the second step. In the third step, the results collected during the simulations are examined and analyzed. Based on the analysis of these results, the initial system is modified accordingly in order to correct possible errors or to add more functionality, which becomes the fourth step in the process. As shown in the figure 7.1, steps 2, 3 and 4 are continuously repeated until satisfactory and correct results are collected [27].

Network simulation is an important tool for researchers and engineers that allow an-

alyzing network's behavior and performance. Network simulator is usually the first place to verify a new idea or existing one. There are various reasons for using network simulators such as lacking hardware, difficulties in modifying the equipment, complexity of real-world equipment, dependable statistics collection and the requirement of separating simulated process from other activities [28]. In this thesis, OPNET IT



1. Create a network model.
2. Choose Statistics to collect from each network object or from the whole network.
3. Execute a simulation.
4. View and analyze results.

Figure 7.1: The Steps Used to Build a Network Model and Run Simulations

Guru Academic Edition 9.1 simulation tool is used. OPNET IT Guru Academic Edition provides a graphical environment that directly reflects the results of actual run, devices, protocols and applications. It gives us a rise for network analysis through powerful tools that is integrated with Graphical User Interface (GUI) [27].

There is another network simulator in the community. It is so called NS2. It was developed under VINT (Virtual InterNetwork Testbed) project, by University of California at Berkeley, University of Southern California's Information Sciences Institute, Lawrence Berkeley National Laboratory (LBNL), and Xerox Palo Alto Research Center (PARC). The main sponsors are the Defense Advanced Research Projects Agency (DARPA) and the National Science Foundation (NSF). Ns-2 seems to be completely free for both educational and commercial purposes (although some older code, still in use, explicitly grants rights to educational type of use only, so the actual status is a bit unclear for commercial projects). The simulator is available with a full source

code, validation tests, a rich set of examples and a good manual. Moreover, additional support may be provided by the ns-2 user mailing list.

The differences between NS and OPNET are enumerated as [28, 29]:

1. In NS, there is a lack of user interface. OPNET has rapid simulation development environment with GUI support that gives opportunity to get what you see without sacrificing from performance.

2. NS is more difficult to learn but OPNET has good documentation. and enriched with exact utilization of real life.

3. NS is less generic and less configurable than OPNET. OPNET has flexibility in model development enhanced with API

4. OPNET has a clear state of mind with illustrative animation support.

5. OPNET is very easy to install and has support for a server running the authorization software

6. OPNET has expanding user community that appears to be dominant supported by forum in the community.

7. OPNET has emerging technologies like discrete event (both fully parallel and serial), flow- based, hybrid simulation, and co-simulation technologies.

Opnet IT Guru Academic Edition System Requirements:

- Intel Pentium III, 4 or compatible (500 MHz or better)

- 256 MB RAM
- 400 MB disk space
- Display: 1024 x 768 or higher resolution, 256 or more colors
- The English language version of the following operating systems are supported: Microsoft Windows NT (Service Pack 3, 5, or 6a; Service Packs 4 and 6 are not supported) Windows 2000 (Service Pack 1 and 2 are supported but not required) Windows XP (Service Pack 1 is required)

7.2 Incorporating RSVP with OPNET

7.2.1 Data Colleciton Specification

OPNET allows for a very large number of potential statistics. For this reason, collection mechanisms are deactivated by default when a simulation is executed. However, OPNET provides a mechanism to explicitly activate statistics of particular interest, which are recorded in appropriate output files. This is accomplished by specifying a list of probes when running a simulation, which indicate the particular statistic that should be collected.

In order to investigate the RSVP in the network environment, the following statistics have been specified and studied that allowed for the system's behavioral study and validation.

- Link Delays, Throughput and Utilization (for identifying any congested links).

- RSVP Control Traffic sent and received (for verifying the protocol's correct functionality).
- Application Traffic sent and received (for verifying the correct traffic flow through the system's components).

7.2.2 QOS Setup And Specification

Quality of Service parameters setup may be performed at the IP level. At the IP level QoS is achieved by the utilizing the RSVP protocol.

The following sections will describe how QoS setup is achieved in the OPNET simulation tool.

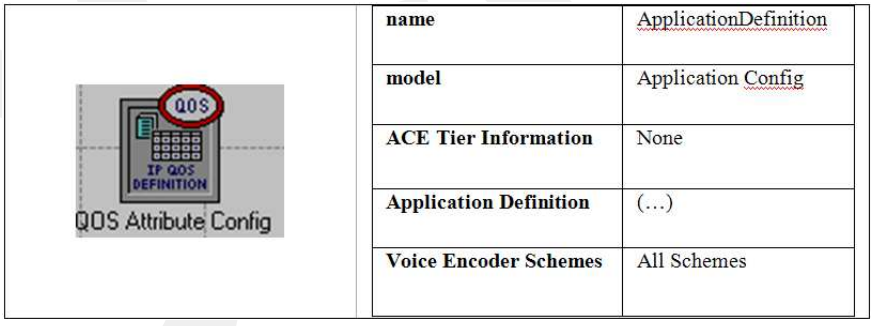
7.2.2.1 IP Level QoS Configuration

Nodes in the network are capable of running TCP or UDP applications, they utilize the Resource Reservation Protocol to request specific Quality of Service from the network for the application data streams that are serving and are sensitive to time delay.

In essence, RSVP reserves resources (bandwidth and buffers) for the traffic, which in turn should result in lower delay and delay variation, and eventually to improved system throughput. OPNET allows configuration of RSVP on intermediate and advanced host and router node models, which means models whose names end with "_int" or "_adv".

The first step in configuring RSVP is to create named data-flow specifications. This is achieved by including a "QoS Attribute Configuration" object in the scenario (see Figure 7.2), which may be found in the "utilities" object palette. Data-flow specifications can be created by editing the RSVP Flow Specification attribute of the QoS Attribute Configuration objects (see Figure 7.2). These specifications are used when defining RSVP profiles and for reserving bandwidth and buffer size in a node's IP links.

The second step in the RSVP configuration involves the creation of the RSVP profiles. This may be performed by editing the RSVP profiles attribute of the QoS Attribute Configuration object. By doing so, the various types of RSVP reservations that may be used in the scenario are defined. In addition to a specific QoS, each profile defines if, how and when a receiving node requests a reservation based on the following attributes: A host application requests a reservation only if a PATH message indicates



name	ApplicationDefinition
model	Application Config
ACE Tier Information	None
Application Definition	(...)
Voice Encoder Schemes	All Schemes

Figure 7.2: QoS Attribute Configuration Object and Its Attributes

a traffic load higher than that defined by the Threshold (bytes/sec) attribute.

The Reservation Style attribute may be used for setting fixed- filter, shared-explicit or wildcard-explicit reservations.

Specific data flows (defined in the QoS Attribute Configuration object's RSVP Flow Specification attribute) and sender list may be assigned to the RSVP profile by editing the Reservation Parameters attribute

If the network cannot grant a reservation request, the RSVP profile's Retry Policy attribute determines how the host responds.

In the third step of the RSVP configuration process, the RSVP protocol is enabled in all of the intermediate nodes. By default, RSVP is disabled on all node models.

Consequently, each intermediate node's IP Address Information. QoS Info. RSVP Info attribute must be set to "Enabled". Due to the fact that modeling RSVP results in higher packet generation and simulation run-times, it is recommended that RSVP be enabled only on the nodes directly related to the study of each scenario.

After enabling RSVP in all intermediate nodes, RSVP must be enabled in receiver applications running on host nodes.

Each host that runs RSVP must have the application RSVP parameters. Application:RSVP Parameters.<Application name> .

RSVP Status attribute enabled. In addition, after enabling RSVP, one or more RSVP profiles must be assigned to the application. These are used by the node to determine the type of reservation it will request from the network upon receiving a PATH message.

Finally, data flows for an RSVP-enabled application session must be specified. This

task is performed by including an Application Definition Configuration object in the scenario (Figure7.3), which may also be found in the Utilities object palette, and editing the object's Application name. RSVP Parameters attribute. When editing this attribute, the RSVP Status must be set to "Enabled" and both the **Outbound Flow** and **Inbound Flow** attributes must be assigned data flows. These last two attributes specify the bandwidth and buffer size that will be reserved on the node's IP links for the specified application's traffic [27].

name	ApplicationDefinition
model	Application Config
ACE Tier Information	None
Application Definition	(...)
Voice Encoder Schemes	All Schemes

Figure 7.3: Application Definition Configuration Object and Its Attributes

Resource Reservation Event Sequence is described below: When a simulation is executed and RSVP is enabled in particular nodes in the network model, the following events happen, which eventually establish the resource reservation:

1. When a new TCP or UDP session is established, the Application/RSVP Interface process sends a request to the local RSVP process to start RSVP processing for the session (the RSVP/Application Interface process is a process that is responsible for initializing the RSVP processing for a session).

2. Upon receiving the request, the local RSVP process starts sending RSVP signaling PATH messages towards the receiver.

3. The RSVP process of the receiver receives the PATH message and forwards the

information from the message to the Application/RSVP interface process.

4. The Application/RSVP interface process decides whether the receiver will request a reservation for the traffic it receives. If it decides to make a reservation, it contacts the local RSVP process.

5. The RSVP process at the receiver creates a RESV message, which is sent towards the sender.

6. The RESV message is forwarded to the RSVP process at each intermediate node (despite the fact that the packet's final destination is the sender). The RSVP process makes a reservation request to the local Traffic Control process (the Traffic Control process is a process that is responsible for packet queuing and output scheduling on a node, by using techniques such FIFO, WFQ, Custom Queuing, etc).

7. The Traffic Control process receives a request from RSVP. Based on the available resources, it grants or denies the reservation.

8. If the reservation is granted, the RESV message is forwarded towards the sender. Otherwise, the RESV_ERR message is sent to the receiver.

9. Steps 6, 7, and 8 are repeated at each intermediate node.

10. Finally, the RESV message reaches the sender. The sender sends a RESV_CONFIRM message that informs the receiver that the reservation has been established.

7.2.2.2 Profile Configuration

OPNET utilizes a generic network application model to generate typical traffic patterns. This is the applications model and most of the standard OPNET node models embed the applications model to model the traffic they generate and the way they treat the traffic they receive [30, 31]. Each application can be enabled or disabled on the client nodes and can also be specified as a supported application service type on the server nodes. Applications were specified in "Application Definition Configuration" objects (see Figure 7.3) at each level of the factory, by editing the Application Definitions attribute. Each application was named after the network level it was executing. In addition, an increasing number was appended to that name in order to differentiate the same applications that were executing in different factories, sections and manufacturing cells. After specifying the applications, profiles for each application had to be created, in order to allow the node models to utilize them. This was performed by first including a "Profile Definition" object in the same level as the application (see Figure 7.4), which may be found in the "utilities" object palette, and then editing the Profile Configuration attribute of this object.

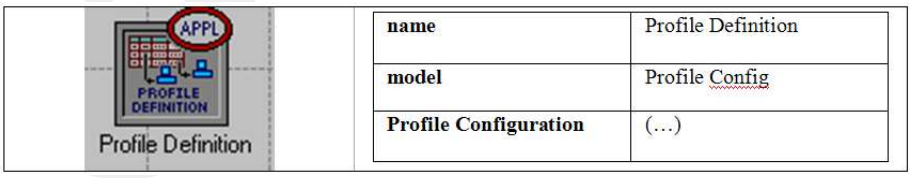


Figure 7.4: Profile Definition

CHAPTER 8

EXPERIMENTS AND RESULTS

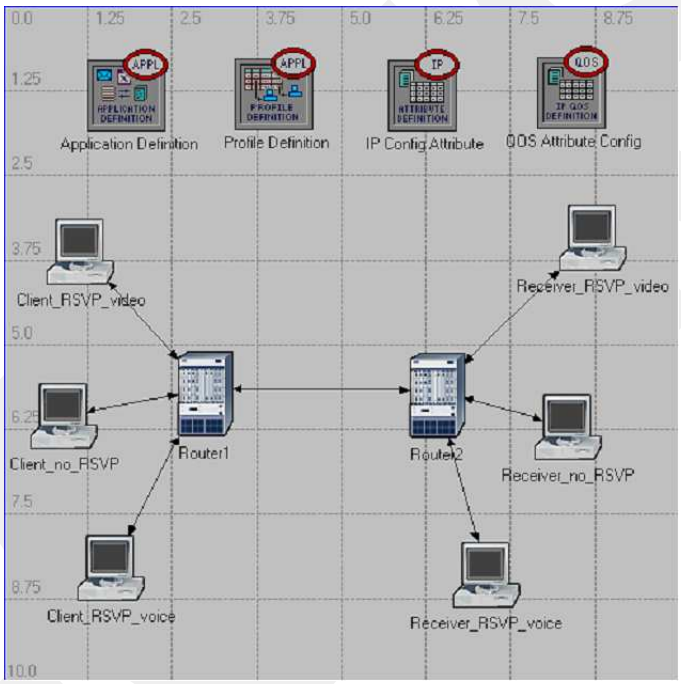


Figure 8.1: RSVP Enabled Network

In order to evaluate the performance of the RSVP model, this network model has been created. The Ethernet network consists of three clients sending traffic to associated receivers via routers. All the nodes in the network are connected with PPP_DS0 links with a 64 Kbps data rate. Client_RSVP_video node and Client_no_RSVP video node are video applications while Client_RSVP_voice node is voice application. Two video conferencing sessions are competing for the same resources. Traffic between

Client_RSVP_video and Receiver_RSVP_video uses RSVP to reserve resources also traffic between Client_RSVP_voice and Receiver_RSVP_voice uses RSVP to reserve resources while traffic between Client_no_RSVP and Receiver_no_RSVP uses best effort service. There is one session for each source-destination pair in-place for the duration of the simulation. The reservation will be made for traffic in both directions. The traffic generated by each client is described as having a bandwidth of 5,000 bytes/sec and a buffer size of 5,000 bytes. These parameters will be used for the reservation. Also reservation style is selected Wild Card. OPNET supports five different QoS policies: RSVP Protocol, Committed Access Rate, (CAR), Custom Queuing (CO), Priority Queuing (PQ), and Weighted Fair Queuing (WFQ). In addition OPNET's RSVP model supports Controlled Load service. This service is supported for WFQ and Custom Queuing schemes. This scenario based on WFQ and RSVP.

Configuring Applications

Attributes describing RSVP parameters set by the application are defined in two objects: the QoS Attribute Configuration object and the Application Attribute Configuration object. To run an RSVP simulation, both objects must be included in the scenario.

Also Profile Definition Attributes and IP Configuration Attributes can be seen for the general network configuration.

Figure 8.2 illustrates the Profile definition Attributes are video_reserved, Video_unreserved and voice_reserved which are defined in Profile Definition. Figure 8.3 shows the IP Configuration Attributes with default values.

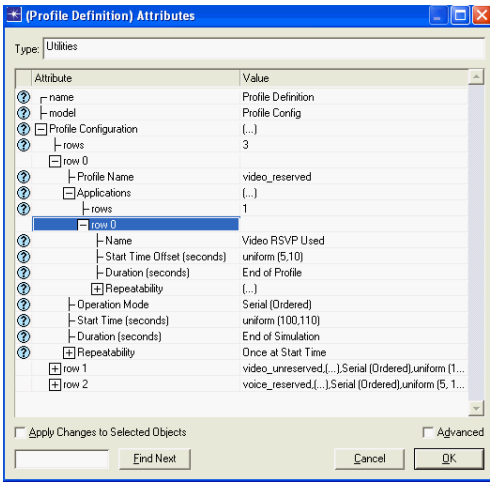


Figure 8.2: Profile Definition Attributes

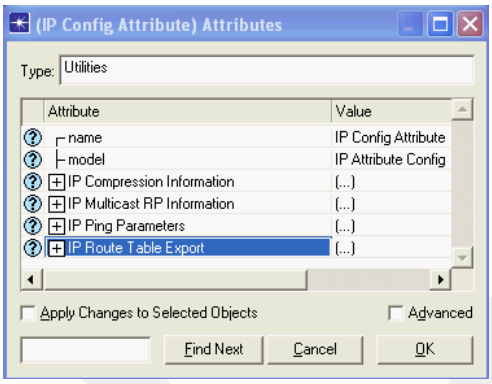


Figure 8.3: IP config Attributes

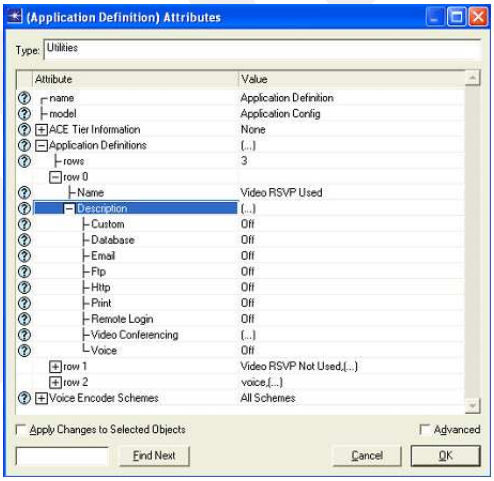


Figure 8.4: Application Definition Attributes

Figure 8.4 shows the Video conferencing and voice applications are defined in Application Definition. For Video Applications, video conferencing status are "On" and other applications are "Off". For voice Application, voice status is "On" and other applications are "Off".

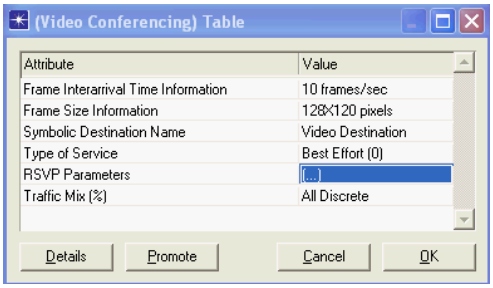


Figure 8.5: Video Conferencing Table

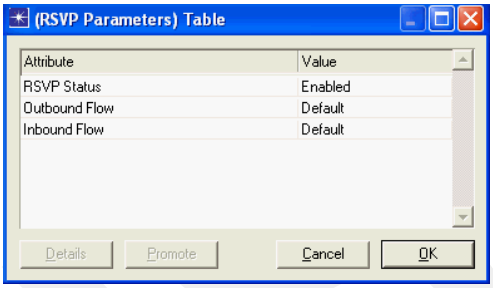


Figure 8.6: RSVP Parameters Table

The following figures 8.5 and 8.6show the RSVP parameters on Video Conferencing application and RSVP status of this application.

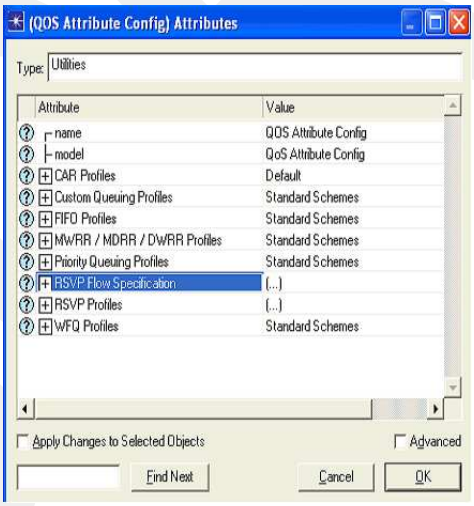


Figure 8.7: QoS Attribute Config Attributes

Figure 8.7, Figure 8.8, and Figure 8.9 illustrate the RSVP Flow Specification and RSVP Profiles. The traffic generated by each client is described as having a bandwidth of 5,000 bytes/sec and a buffer size of 5,000 bytes. These parameters will be used for the reservation. Also reservation style is selected Wild Card.

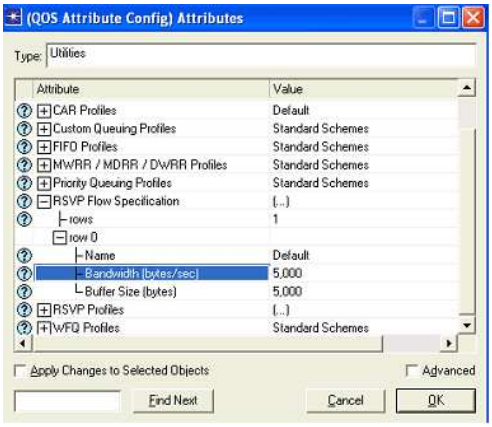


Figure 8.8: QoS Flow Spec Atribute

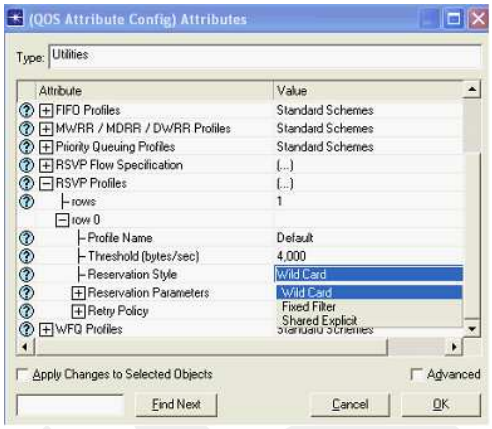


Figure 8.9: QoS RSVP Reservation Style

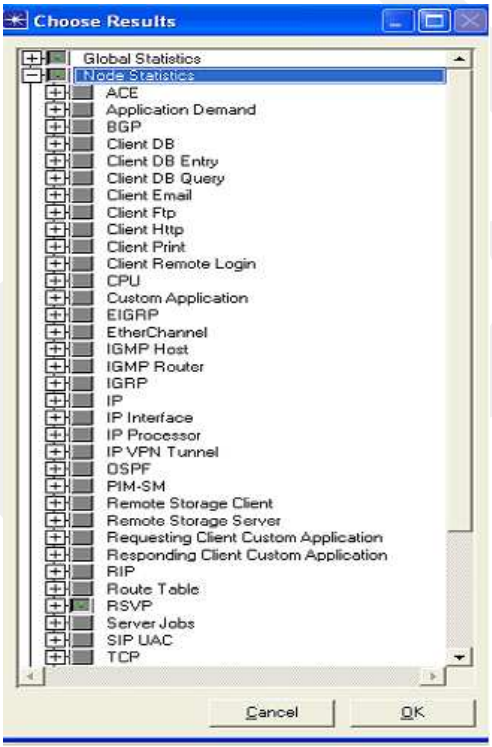


Figure 8.10: Statistics Collection

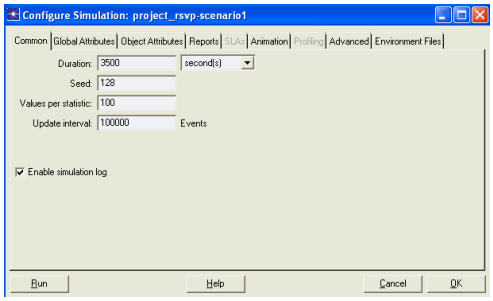


Figure 8.11: Simulation Runtime Settings

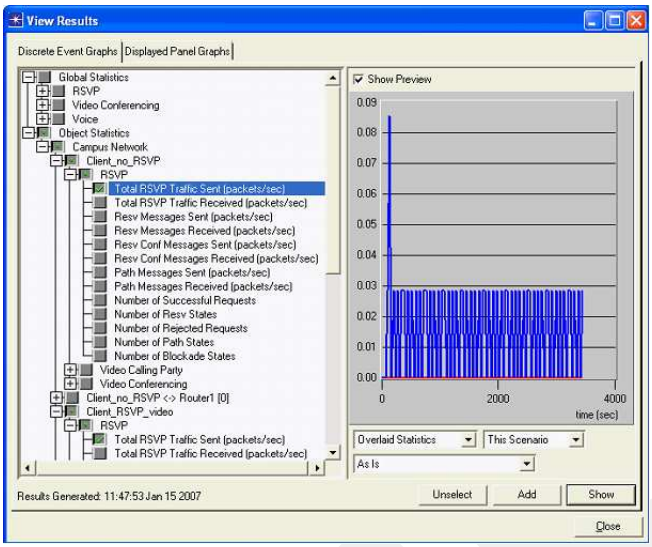


Figure 8.12: Selected Simulation Results

Global and Node Statistics are selected before running simulation in Figure 8.10. RSVP Global Statistics capture the total amount of RSVP traffic sent and received in the whole network. RSVP Node Statistics capture the nodes and links individually. Figure 8.11 illustrates The Simulation Configuration. Figure 8.12 shows the results for selected information after running simulation.

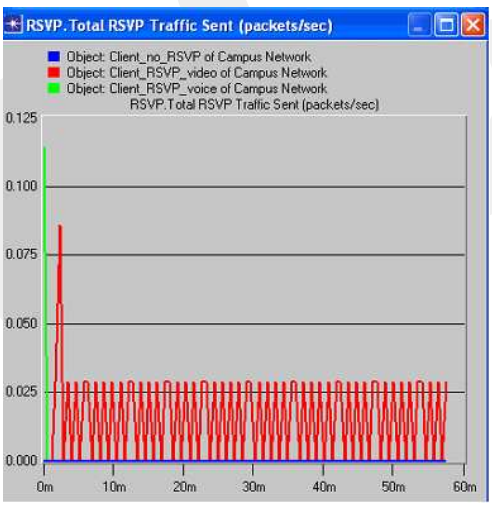


Figure 8.13: Tot. Rsvp Traff. Sent

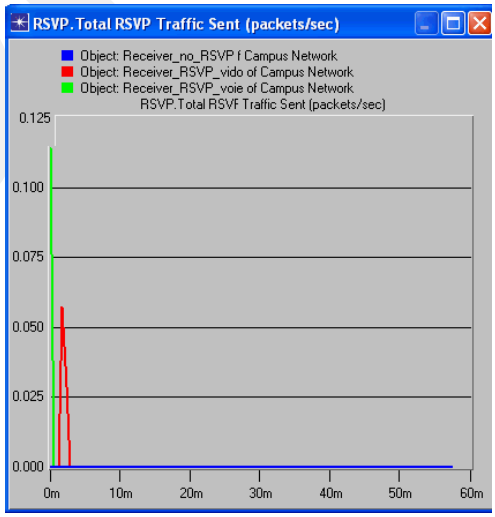


Figure 8.14: Tot. Rsvp Traff. Sent

Figure 8.13 and Figure 8.14 show the Total RSVP sent packets on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using

RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

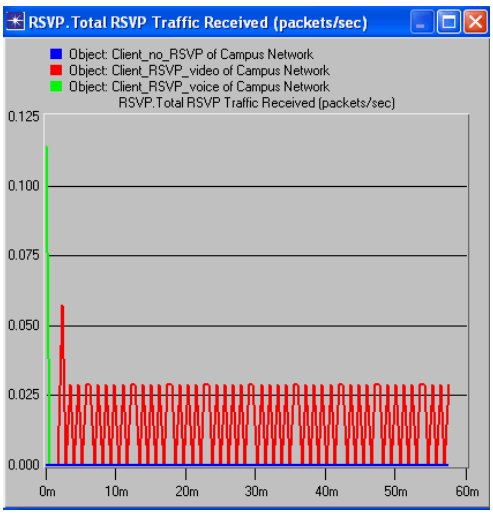


Figure 8.15: Tot. Rsvp Traff. Recv

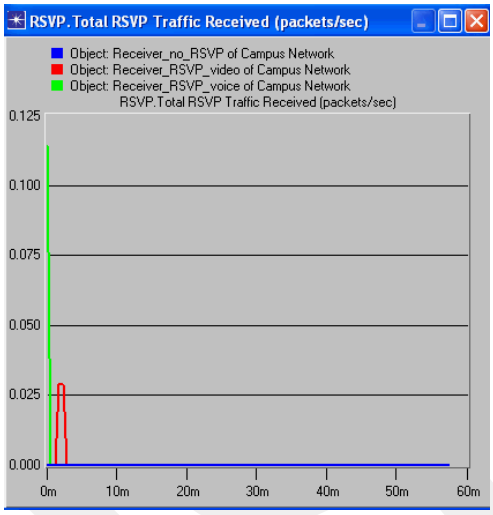


Figure 8.16: Tot. Rsvp Traff. Recv

Figure 8.15 and Figure 8.16 show the Total RSVP received packets on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

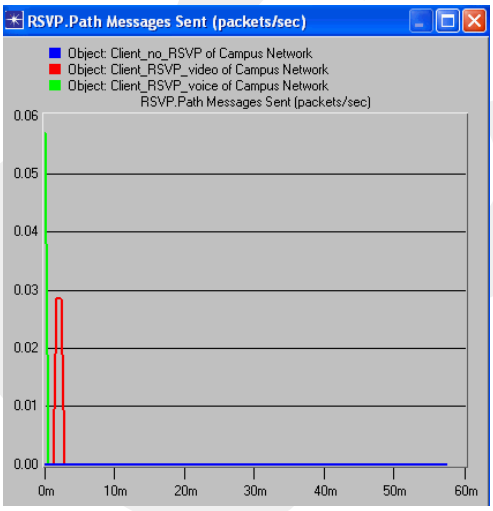


Figure 8.17: Rsvp Path Msg Sent

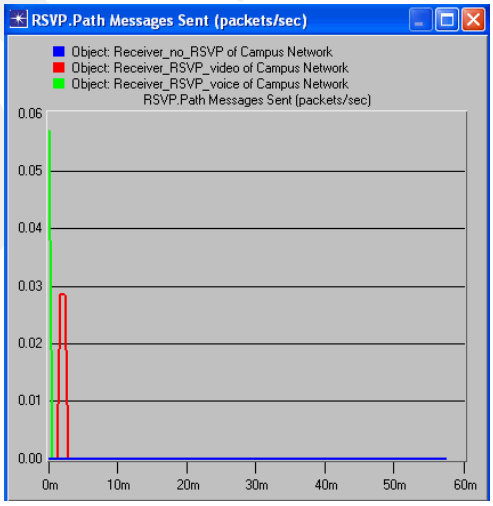


Figure 8.18: Rsvp Path Msg Sent

Figure 8.17 and Figure 8.18 show the RSVP Path Messages Sent on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

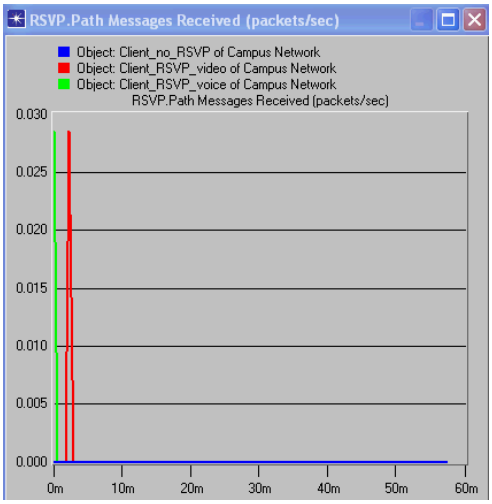


Figure 8.19: Rsvp Path Msg Recv

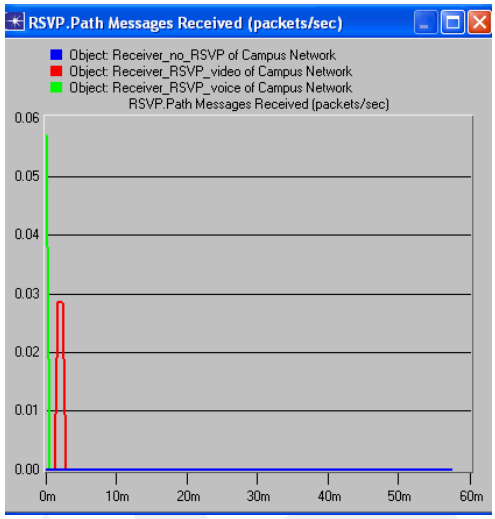


Figure 8.20: Rsvp Path Msg Recv

Figure 8.19 and Figure 8.20 show the RSVP Path Messages Received on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

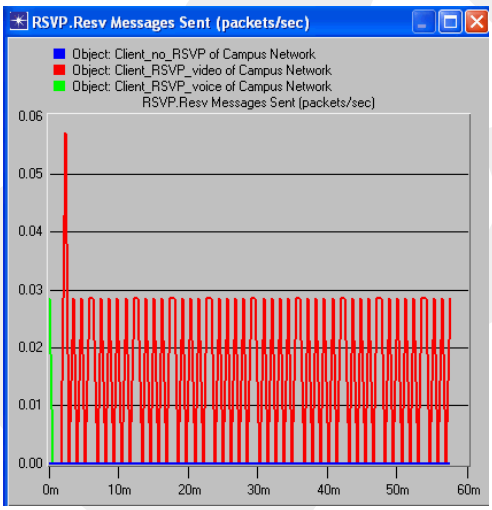


Figure 8.21: Rsvp Resv Msg Sent

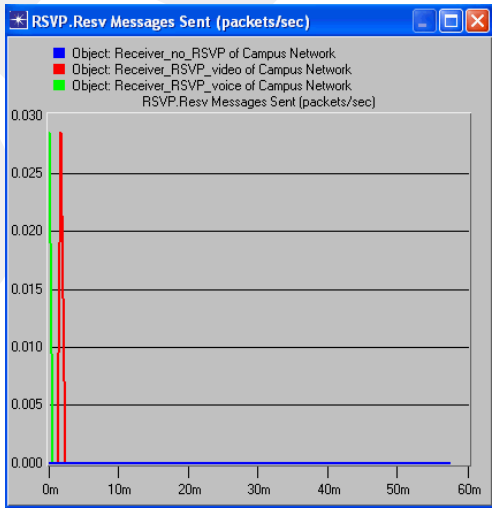


Figure 8.22: Rsvp Resv Msg Sent

Figure 8.21 and Figure 8.22 show the RSVP Resv Messages Sent on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

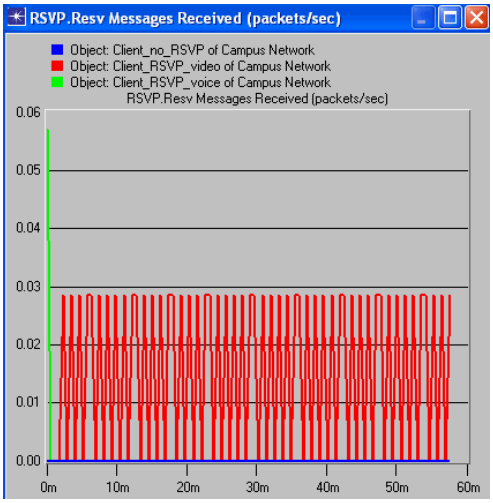


Figure 8.23: Rsvp Resv Msg Recv

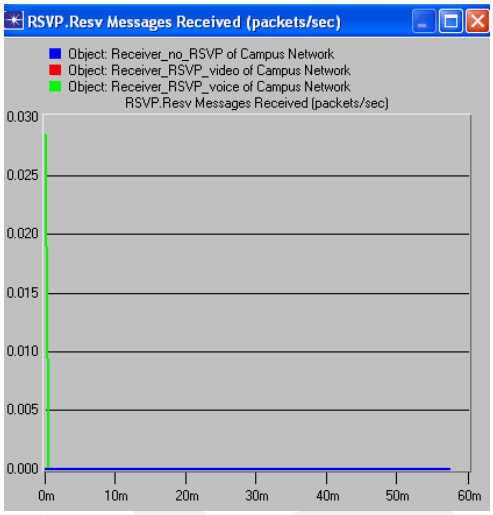


Figure 8.24: Rsvp Resv Msg Recv

Figure 8.23 and Figure 8.24 show the RSVP Resv Messages Received on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

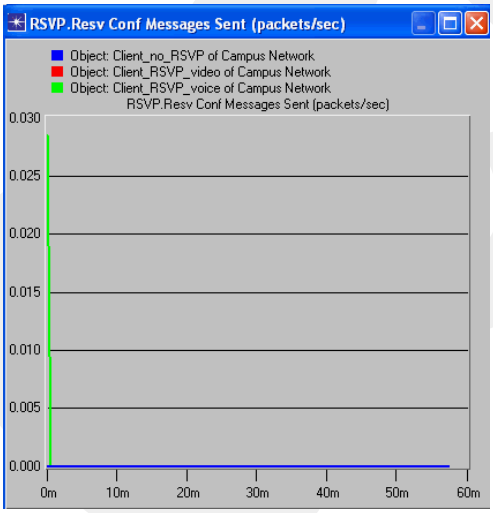


Figure 8.25: Rsvp Conf Msg.s Sent

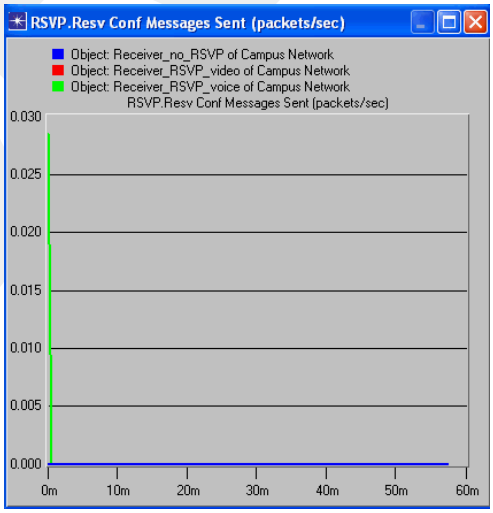


Figure 8.26: Rsvp Conf Msg.s Sent

Figure 8.25 and Figure 8.26 show the RSVP Conf Messages Sent on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

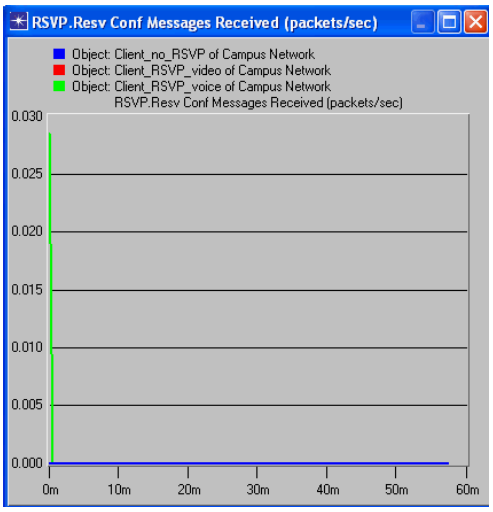


Figure 8.27: Rsvp Conf Messages Recv

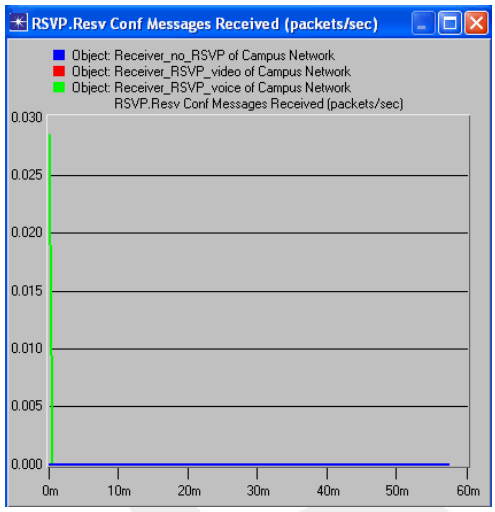


Figure 8.28: Rsvp Conf Messages Recv

Figure 8.27 and Figure 8.28 show the RSVP Conf Messages Received on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

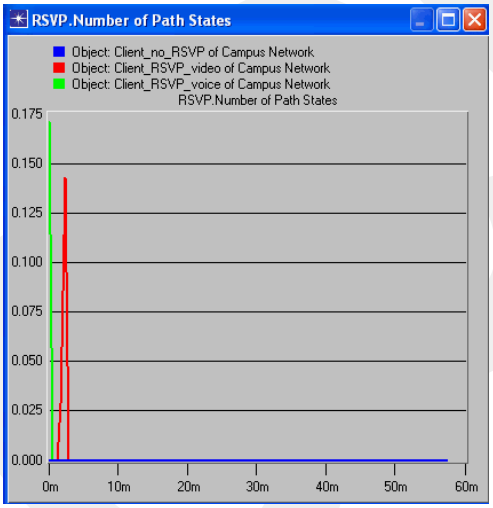


Figure 8.29: No of Path States

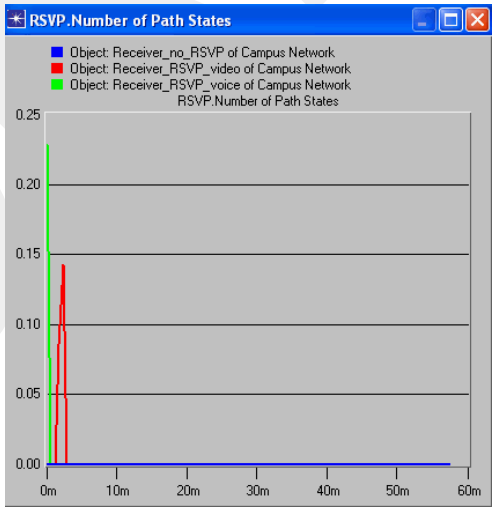


Figure 8.30: No of Path States

Figure 8.29 and Figure 8.30 show the RSVP Number of Path States on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero, because Path State information is used in RSVP.

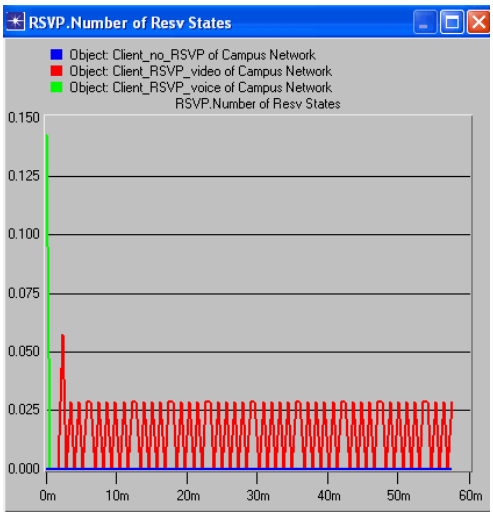


Figure 8.31: No of Resv States

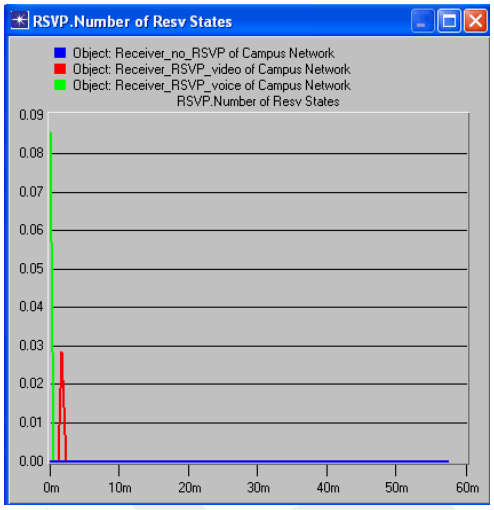


Figure 8.32: No of Resv States

Figure 8.31 and Figure 8.32 show the RSVP Number of Resv States on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero. Because Resv state information is used in RSVP.

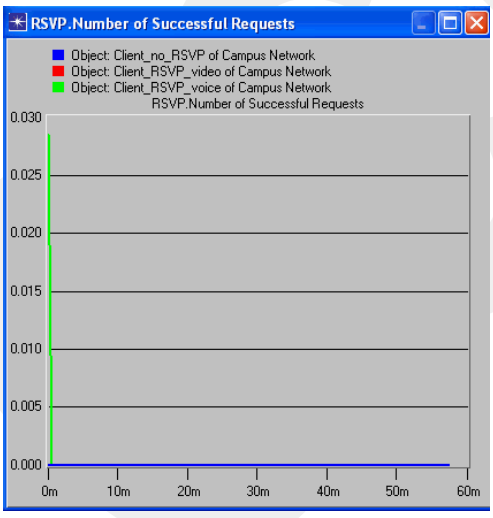


Figure 8.33: No of Succ. Req.s

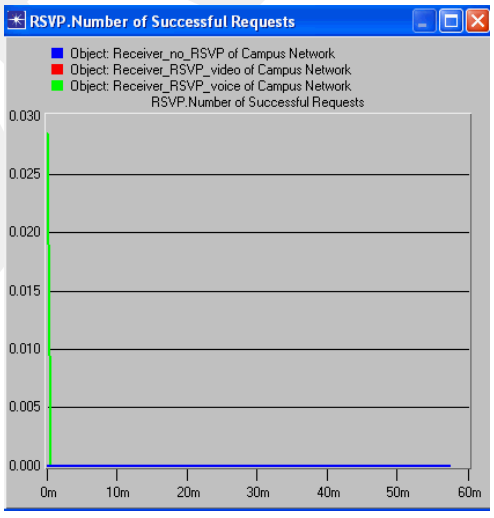


Figure 8.34: No of Succ. Req.s

Figure 8.33 and Figure 8.34 show the RSVP Number of Successful Requests on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP (Client_no_RSVP node and Receiver_no_RSVP node) experienced zero.

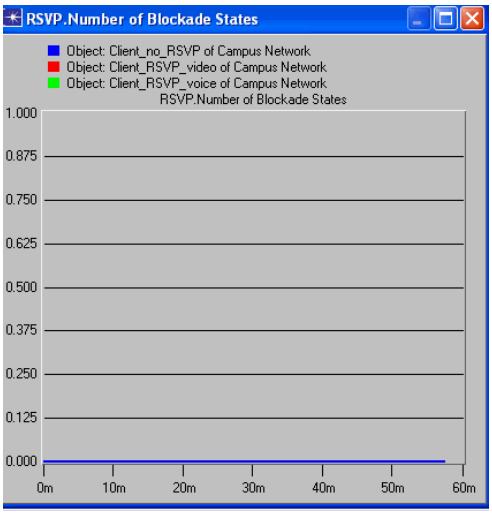


Figure 8.35: Blockade States

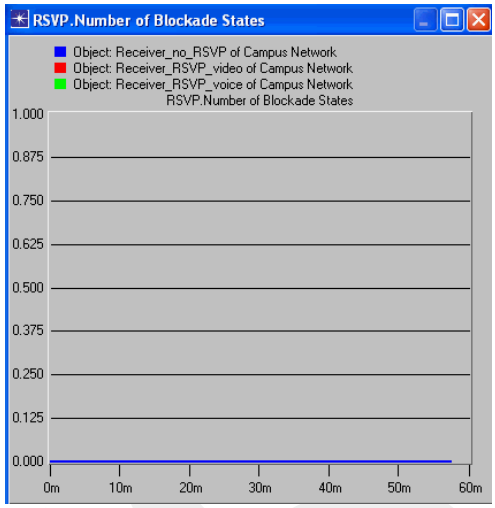


Figure 8.36: Blockade States

Figure 8.35 and Figure 8.36 show the RSVP Number of Blockade States on Clients and Receivers experienced using RSVP and not using RSVP. As expected, traffic not using RSVP and traffic using RSVP experienced zero, because Blockade State information is used in RSVP routers.

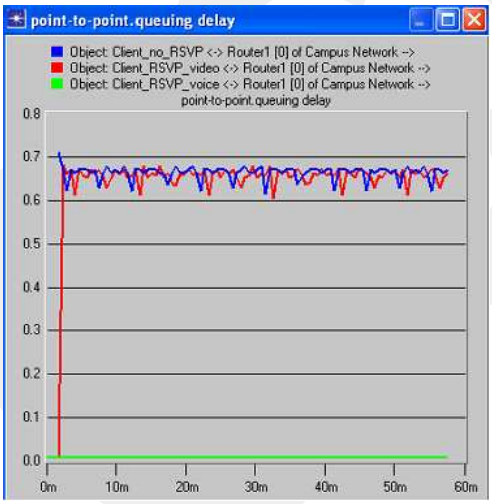


Figure 8.37: P2P Queuing Delay

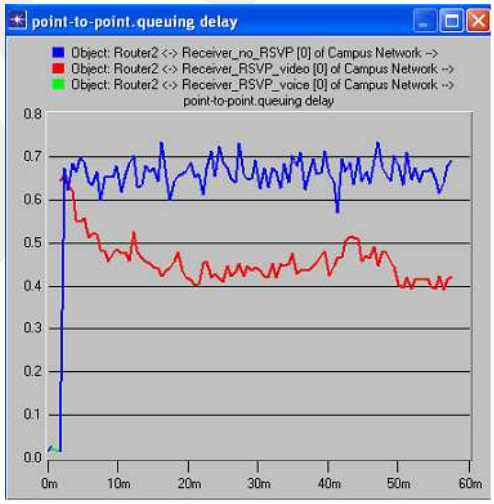


Figure 8.38: P2P Queuing Delay

Figure 8.37 and Figure 8.38 compare the point-to-point queuing delay experienced using RSVP with the queuing delay experienced not using RSVP between the Clients to Router1 and Router2 to Receivers. As expected, traffic using RSVP reservation

experienced less queuing delay.

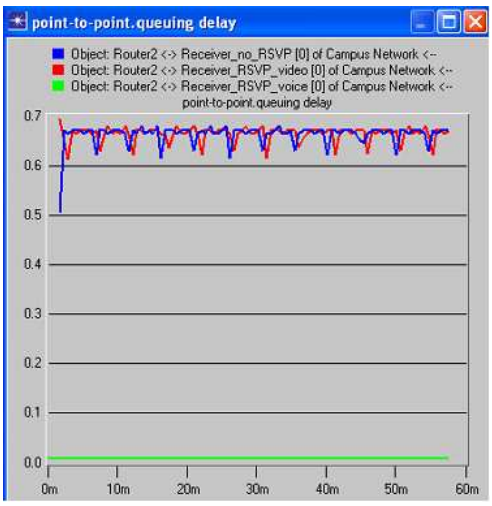


Figure 8.39: P2P Queuing Delay

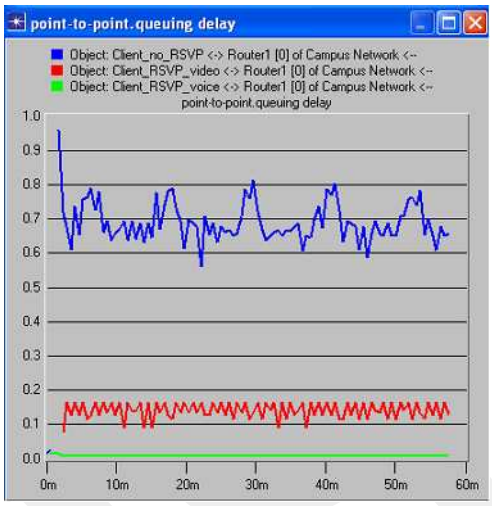


Figure 8.40: P2P Queuing Delay

Figure 8.39 and Figure 8.40 compare the point-to-point queuing delay experienced using RSVP with the queuing delay experienced not using RSVP between the Receivers to Router2 and Router1 to Clients. As expected, traffic using RSVP reservation experienced less queuing delay.

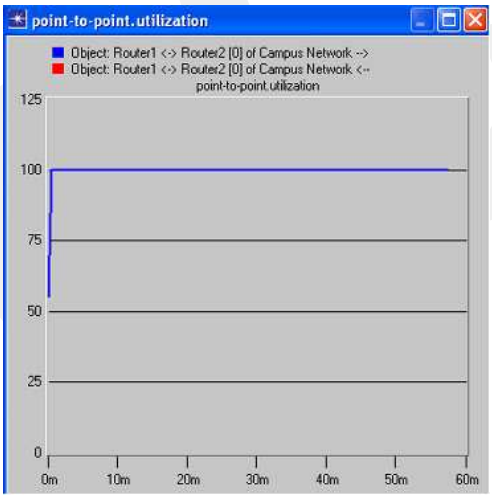


Figure 8.41: Link Utilization

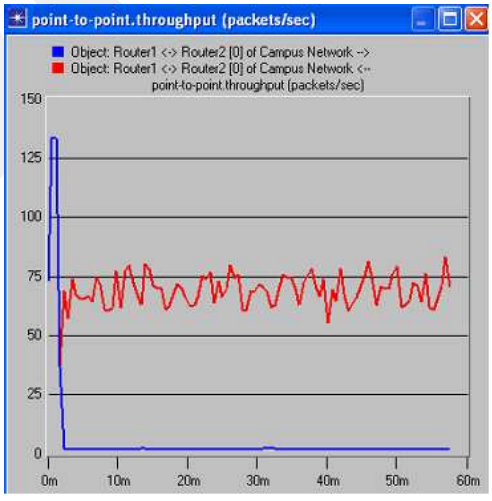


Figure 8.42: Link Utilization

In Figure 8.41, utilization of the link between Router1 and Router2 is shown. RSVP is used between Router1 and Router2. Maximum Reservable Bandwidth is a percent-

age of the link bandwidth that RSVP can use. Maximum Reservable Bandwidth is configured to 100%. As expected, traffic using RSVP reservation experienced using full bandwidth. In Figure 8.42, throughput of the link between Router1 and Router2 is shown. It clearly shows how sufficient the link between Router 1 and Router 2 is for this load.

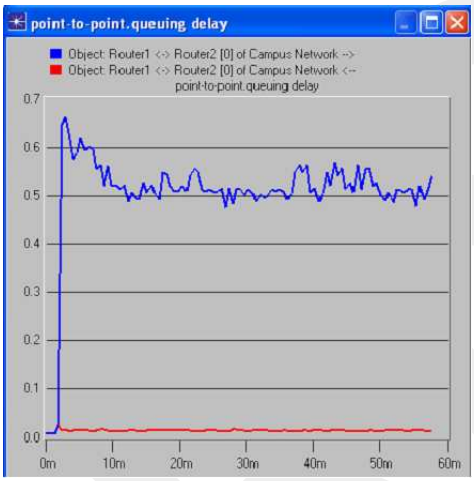


Figure 8.43: Link Queuing Delay

In Figure 8.43 queuing delay of the link between Router1 and Router2 is shown. RSVP is used between Router1 and Router2. In my scenario all clients are connected to Router1. Therefore Router1 sends more confirmation messages. As expected, the outgoing link between Router1 and Router2 experienced higher queuing delay.

Figure 8.44 and Figure 8.45 show the Total RSVP Traffic Sent and Received on Router1 and Router2 experienced using RSVP. As expected, the network is used by Router2 all the time so there is no too much change. The Router1 uses the network some times.Therefore fast change can be seen.

These two figures(Figure 8.46 and 8.47) show the Total RSVP Resv Messages Sent and Received on Router1 and Router2 experienced using RSVP. As expected, the

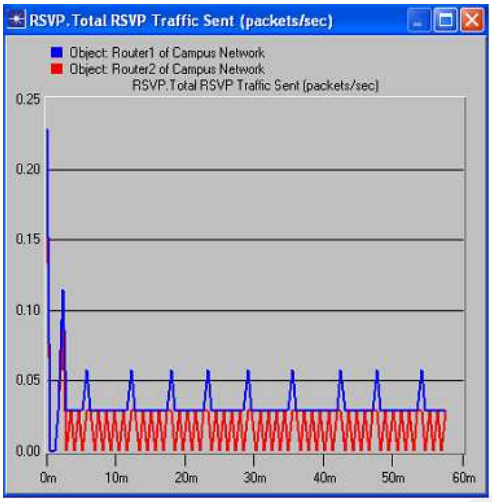


Figure 8.44: RSVP Traffic Sent

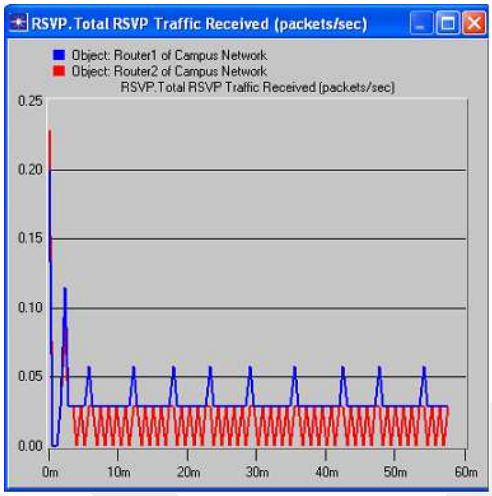


Figure 8.45: RSVP Traffic Recv

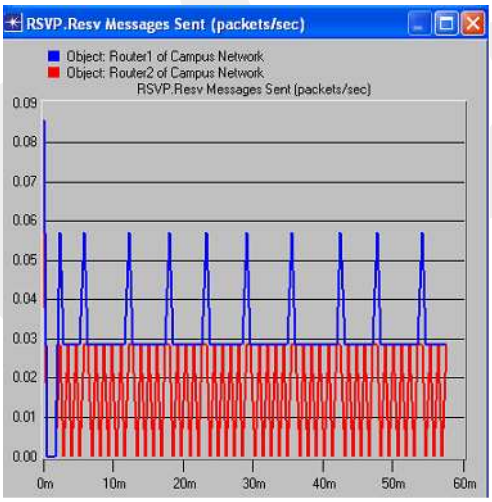


Figure 8.46: RSVP Resv Messages Sent

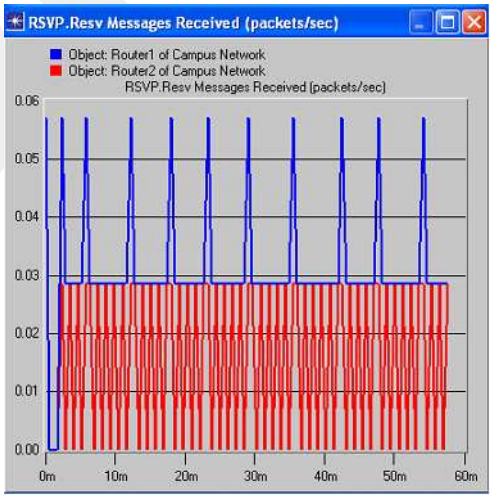


Figure 8.47: RSVP Resv Messages Recv

network is used by Router2 all the time so there is no too much change. The Router1 uses the network some times.Therefore fast change cen be seen.

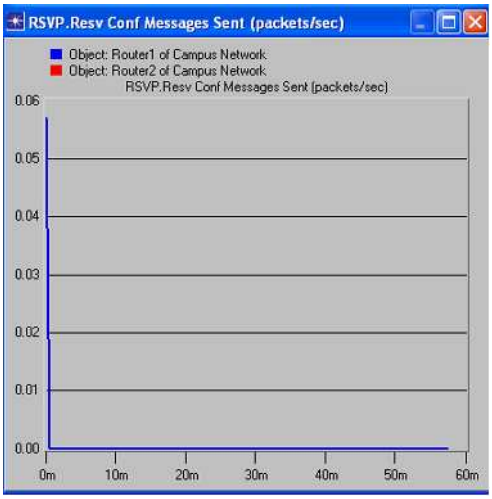


Figure 8.48: RSVP Resv Conf. Sent

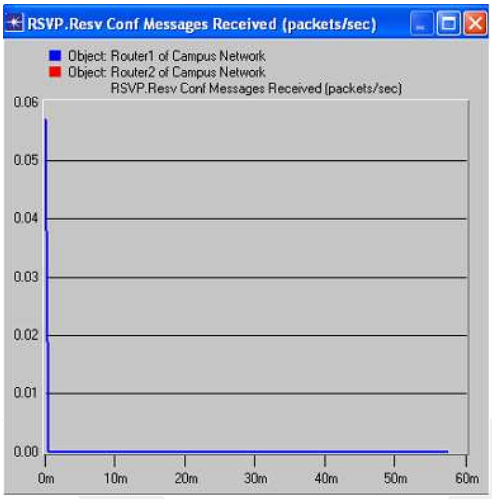


Figure 8.49: RSVP Resv Conf. Recv

Figure 8.48 and Figure 8.49 illustrate the RSVP Resv Confirmation Messages Sent and Received on Router1 and Router2. In very minute one confirmation message is sent.

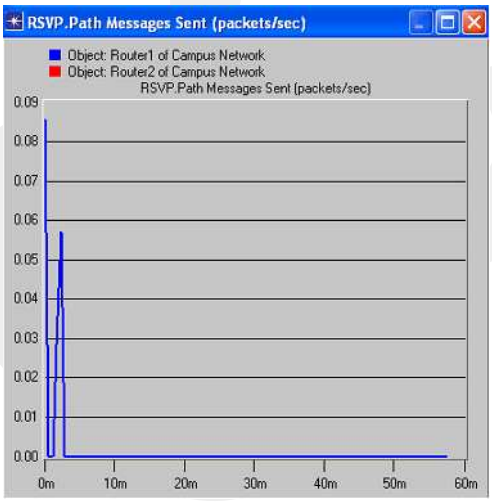


Figure 8.50: RSVP Path Msg Sent

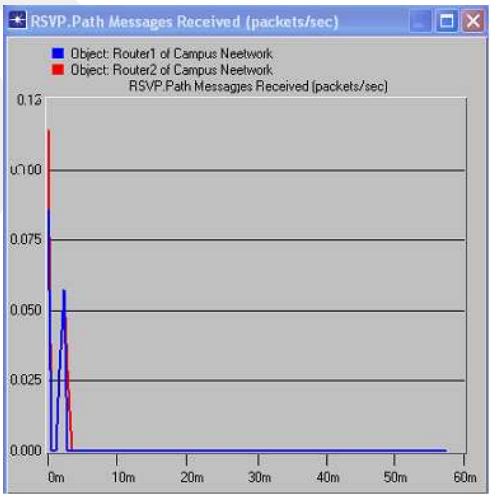


Figure 8.51: RSVP Path Msg Recv

RSVP Path Messages Sent and Received are shown in the Figure 8.50 and Figure 8.51.

Figure 8.52 shows the Number of Successful Requests on Router1 and Router2. RSVP

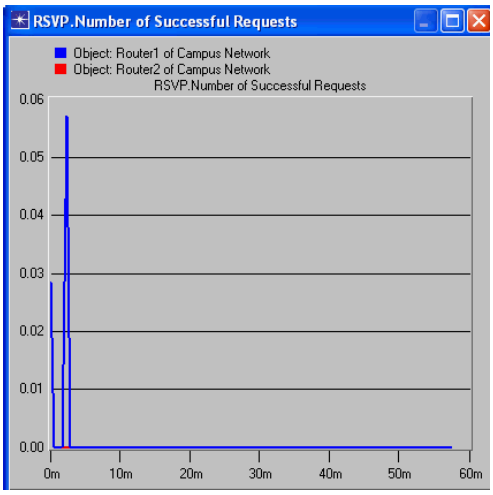


Figure 8.52: Succ. RSVP Requests

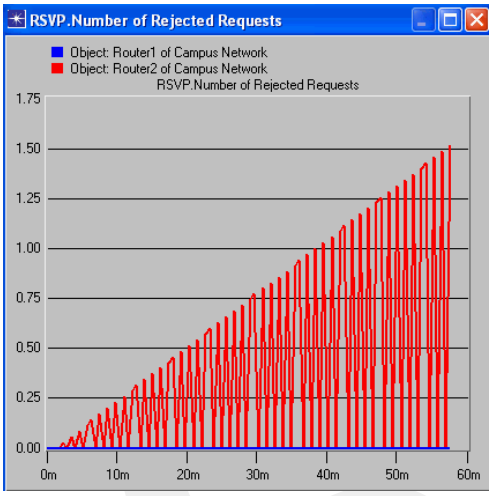


Figure 8.53: Rej. RSVP Requests

setting is made by Router1.Therefore successful requests are available on Router1. Figure 8.53 illustrates the Number of Rejected Requests on Router1 and Router2. Because of the clients, all messages are sent via Router1.Router2 sends responses but all requests from Router1 are successful. Thus number of rejected requests on Router1 is zero.

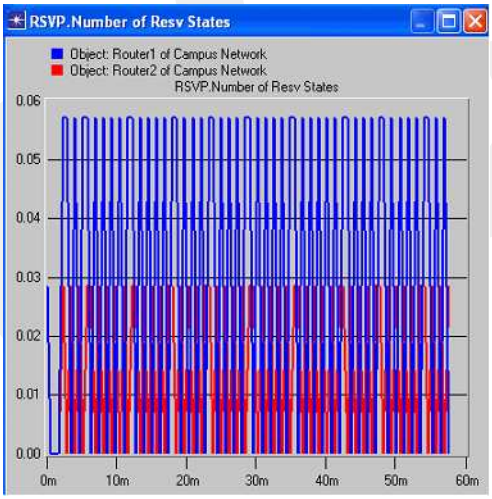


Figure 8.54: RSVP Path States

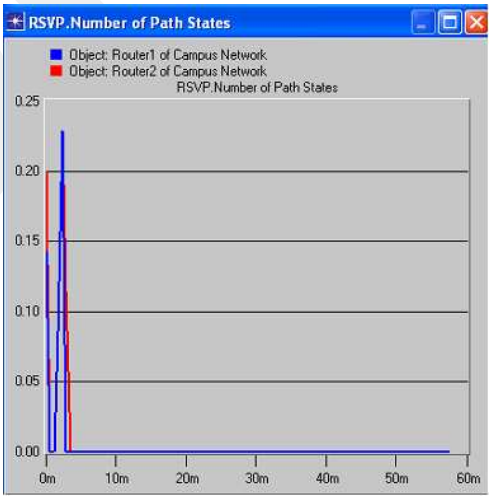


Figure 8.55: RSVP Resv States

Path states are established by Path messages. Number of RSVP Path States on Router1 and Router2 is experienced are shown in Figure 8.54. Figure 8.55 shows

the Total RSVP Resv States on Router1 and Router2 experienced using RSVP. Each receiver periodically sends a Resv message that establishes or updates the reservation state. All the messages are sent via Router1 and requests are resulted from Router 1.Thus higher results can be seen on Router 1.

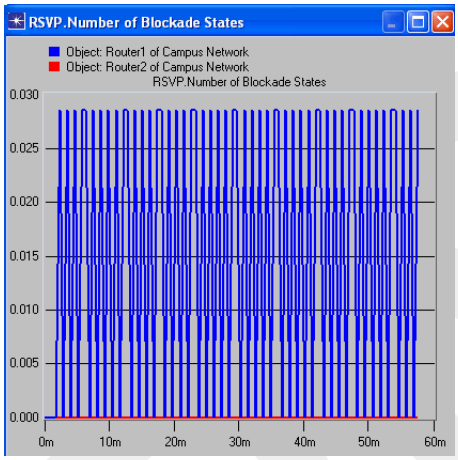


Figure 8.56: RSVP Blockade States

A reservation request that fails Admission Control creates blockade state which can be seen in Figure 8.56. All the messages are sent via Router1 and requests are resulted from Router 1. So blockade states results can be seen on Router 1.

CHAPTER 9

CONCLUSIONS

We conducted some experiments by using OPNET IT Guru Academic Edition 9.1. A scenario is established. Clients and receivers with/without RSVP are used in our simulation. Some results are obtained link utilization, throughput, point to point delay, queuing delay, IP traffic are measured. RSVP protocol does allocation of bandwidth before transmission. If allocation is not done, data transmission does not occur.

In summary RSVP has the following attributes [1]:

- RSVP makes resource reservations for unicast and multicast applications.
- RSVP sessions are simplex. Thus, a bidirectional exchange of data between a pair of machines actually constitutes two separate RSVP simplex sessions.
- RSVP is receiver-oriented. The receiver of a data flow initiates and maintains the resource reservation used for that flow.
- RSVP maintains soft state in routers and hosts, providing graceful support for dynamic membership changes and automatic adaptation to routing changes.
- RSVP is not a routing protocol but depends upon present and future routing

protocols.

- RSVP transports and maintains traffic control and policy control parameters that are opaque to RSVP.
- RSVP provides several reservation models or styles to fit a variety of applications.
- RSVP provides transparent operation through routers that do not support it.
- RSVP supports both IPv4 and IPv6.

APPENDIX

REFERENCES

[1] http://www.ietf.org/rfc2205.txt.

[2] **VOGEL, A**., et. al. (1994), *Distributed Multimedia Applications And Quality of Service: A Survey*, CASCON '94: Proceedings of The 1994 Conference of The Centre for Advanced Studies on Collaborative Research, pp. 71, Toronto, Ontario, Canada, IBM Press.

[3] **TANENBAUM, A. S**. (2003), *Computer Networks*, Prentice Hall, Fourth Edition.

[4] **TSCHUDIN, C.** (2001), *Overiew: Qos and Rsvp*, Data Networks II, Volume 15, pp. 1–29.

[5] **CISCO SYSTEMS, INC**. (2004), Quality *of Service Networking Internetworking Technologies Handbook*, Cisco Press, pp. 49–1:19-32.

[6] http://www.ietf.org/rfc/rfc2211.txt?number=2211.

[7] http://www.ietf.org/rfc/rfc2212.txt?number=2212.

[8] **STALLINGS, W.** (2004), *Computer Networking with Internet Protocols and Technology*, Prentice Hall, NJ.

[9] **BOURAS, C.,** et. al., (2007), *Real-Time Protocols RTP/RTCP, Encyclopedia of Internet Technologies and Applications*, IDEA Group publishing (to appear).

[10] **LIGHT, J.,** et. al. (2004), *Performance Analysis of Audio Codecs over Real-Time Transmission Protocol (RTP) for Voice Services over Internet Protocol*, Communication Networks and Services Research.

[11] **MOON, B., AGHVAMI, H.** (2001), *RSVP Extensions for Real-Time Services in Wireless Mobile Networks*, Communications Magazine, IEEE, Volume 39, Issue 12, pp. 52–59.

[12] **BARZILAI T.** et. al. (1996), *Design and implementation of An RSVP Based Quality of Service Architecture for Integrated Services Internet*, International Conference on Distributed Computing Systems.

[13] **MALLOFRE, G. R.** (2003), *Resource Reservation Protocol (RSVP)*, Seminar on Transport of Multimedia Streams in Wireless Internet, Department of Computer Science University of Helsinki, Finland.

[14] http://www.ietf.org/rfc/rfc2210.txt?number=2210.

[15] http://www.ietf.org/rfc/rfc2747.txt?number=2747.

[16] http://www.ietf.org/rfc/rfc2207.txt?number=2207.

[17] [17] http://www.ietf.org/rfc/rfc2208.txt?number=2208.

[18] **KARSTEN, M.**, et. al. (2001), *Implementation And Evaluation Of The Kom Rsvp Engine*, IEEE Infocom.

[19] http://www.ietf.org/rfc/rfc2961.txt?number=2961.

[20] http://www.ietf.org/rfc/rfc4094.txt?number=4094.

[21] http://www.ietf.org/rfc/rfc1633.txt?number=1633.

[22] **CRAWLEY, E.,** et. al. (1998), *A Framework for QoS based Routing in The Internet, Request for Comments*, Internet Engineering Task Force.

[23] http://www.ietf.org/rfc/rfc2205.txt?number=2205.

[24] **MA, Q.** (1998), *Quality-of-Service Routing in Integrated Services Networks*, PhD thesis, Computer Science Department, Carnegie Mellon University.

[25] [25] **POSTEL, J.** (1981), *Internet Protocol*, STD 5, RFC 791.

[26] http://www.ietf.org/rfc/rfc2475.txt?number=2475.

[27] **ANTONIOU, N. A.** (2001), *Experimental Study of RSVP over ATM*, MSc. Thesis Submitted to San Diego State University.

[28] **MAOWIDZKI, M.** (2004), Network *Simulators: A Developer's Perspective*, Military Communication Institute.

[29] http://www.opnet.com/services/university/opnt_over_ns2.html.

[30] **OPNET.** (2000), *Application Model Description, OPNET General Models Manual*, pp. 1-16.

[31] **HUSTON, G**. (2000), *Internet Performance Survival Guide*, Wiley Computer Publishing.