DEVELOPING QUALITY OF SERVICE MANAGEMENT ARCHITECTURE FOR DELIVERING MULTICAST APPLICATIONS

BY

MARYAM ROSHANAEI

2005

A thesis submitted in partial fulfillment of requirements Kingston University for the degree of Doctor of Philosophy

FACULTY OF TECHNOLOGY

SCHOOL OF COMPUTING AND INFORMATION SYSTEMS

KINGSTON UNIVERSITY

PAGE NUMBERING AS ORIGINAL

ACKNOWLEDGMENTS

This research was supported by the University of Kingston, Faculty of Technology, School of Computing and Information Systems. I would like to express my appreciation for the guidance and suggestions offered by my thesis committee during the course of my research.

Especially, the support and counsel of Ms. Anne Rapley, Director of study, DR. Martin Tunniclif, First supervisor, and Dr. Ron Malyan, Second supervisor. Special appreciation goes to Dr. Shohil khaddaj for his support and encouragement.

I am grateful for support I have received by British Standard Institute (BSI) Study committee 6, Working group 7 and International Telecommunication union (ITU) Study group 17, Working party 2. Special thanks go to Professor Jack Houldsworth, Mr. Jim Long (sadly passed away in November 2003), Professor Seok Joo Koh, Professor ShinGak Kang, Professor Dae Young Kim, Dr. Juyoung Park and the rest of BSI and ITU members.

Special thanks go to my dear sisters, Homa and Shiva, and their families, for the endless joy and support they gave me throughout the years. I am very grateful to my dear friend, Faranak Abtahi, for her support and loving encouragement.

Most of all I wish to express gratitude to my parents for their infinite love, encouragement, and support that they gave me throughout my life: nothing can reward your love.

ABSTRACT

Multicast applications have been a topic of intense research and development efforts over the past couple of years. Both the Internet Engineering (IETF) and International Telecommunication Union (ITU) have been heavily involved in providing quality of service to support multicast application requirements.

Multicast applications have varying performance requirements; therefore it is necessary to design a framework that serves to guarantee quality of services. However the existing best effort services cannot provide the guaranteed service level required by multicast applications.

Two solutions have already been proposed to overcome this problem. The first solution proposed the tree-based functionality approach in the multicast transport protocol providing reliability and scalability between a sender and a group of receivers.

The other solution has proposed end-to-end quality of service (QoS) over the network environment using interoperation of Integrated services (IntServ) and Differentiated services (DiffServ) principles. Both QoS architectures, Integrated and Differentiated services, have their own advantages and disadvantages. With the interoperation of both architectures, it might be possible to build a scalable system, which would provide predictable services. This framework has to be supported by a multicast transport protocol to provide reliability and scalability over the nodes.

The aim of this research is to develop a framework to provide reliably and scalability on nodes (tree-functionality) along with the end-to-end resources, dynamic admission control and scalability over the network (interoperation of IntServ and DiffServ) for multicast applications. The "*Enhanced Communication Transport Protocol*" (ECTP) transport protocol was chosen for this research. ECTP transport protocol is a multicast transport protocol with tree-based functionality to support multicast applications. ECTP transport protocol is also able to provide QoS management functionality established by Integrated or/and Differentiated services to support multicast application. With the QoS management functionality, ECTP

transport protocol could provides reliability and scalability (over nodes) along with end-to-end resource, dynamic admission control and scalability over the network for multicast applications.

This research is focused on the further enhancement and implementation of an ECTP transport protocol, QoS management specification. Two models have been proposed to enable ECTP transport protocol with QoS management functionality established by the IntServ or/and DiffServ principles. Model (I) enables ECTP transport protocol to negotiate end-to-end resource reservation using the standard RSVP (IntServ) signaling protocol. Model (II) enables the ECTP transport protocol to negotiate end-to-end resource reservation using the standard and aggregated RSVP (IntServ and DiffServ) signaling protocol.

The "*Optimized Network Engineering Tool 8.1*" (OPNET) has been used in this research to implement and investigate the ECTP specifications. OPNET simulator provides a comprehensive development environment for modeling and performance of communications networks.

The investigation consists of three case studies. The simulation results have proved that ECTP transport protocol with the tree-based functionality and the QoS management provided by IntServ and DiffServ interoperation produces the best performance for the traffic delay parameter over voice applications.

TABLE OF CONTENT

REFERENCES

APPENDIXES

LIST OF FIGURES

ABBREVIATIONS

ACK	Acknowledgement
AGN	ACK Generation Number
ATM	Asynchronous Transfer Mode
CBT	Core-Base Trees
DVMRP	Distance-Vector Multicast Routing Protocol
NACK	Negative Acknowledgment
ECTP	Enhanced Communication Transport Protocol
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
IP	Internet Protocol
ITU	International Telecommunication Union
ISO	International Standard Organization
OPNET	Optimized Network Engineering Tool
QoS	Quality of Service
RSVP	Resource Reservation Protocol
STD	State Transition Diagram
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SMTP	Simple Mail Transport Protocol
TCP	Transmission Control Protocol
TC	Transport Connection
TAP	Transport Access Point
TS	Transport Service
UDP	User Datagram Protocol
LQA	Lowest Quality Allowed
CHQ	Controlled Highest Quality
OT	Operating Target
MQ	Mean Quality
RAPI	RSVP Application Programming Interface
PIM	Protocol Independent Multicast
ISM	Internet Standard Multicast
SSM	Source Specific Multicast
ITM	Internet Traditional Multicast

CHAPTER 1

INTRODUCTION

1. OVERVIEW

The Internet has grown tremendously over past years. While it was originally intended to be used by the academic and research communities, it has now becomes a common medium for communication and its role in business transactions is expanding daily.

The existing Internet services and protocols provide only best effort service, which only support non-critical applications such as web access and email with limited requirements on delay and bandwidth guarantees.

Since, these applications in existing systems are generally established when they are built, they often do not give users any real influence over the quality of service (QoS) they can expect.

However, emerging next generation applications [Rutkowski99] such as multicast applications, they require diverse Quality of Service (QoS) specifications over the network. These requirements involve simplex or multicast communications and reservations of bandwidth resources. Multicast applications and their users can differ enormously in their service requirements and the resources available to them. Therefore there is an increasing need for modified services that can be tailored for the end users' specific requirements.

One of the existing solutions to overcome this problem the use of multicast transport protocols that support multicast applications requirements. Scalability and reliability are two major features that multicast transport protocols should have to support multicast applications. There have been proposed several approaches to achieve scalability and reliability, and it has been shown that the "tree-based" approach has the best results.

The other existing solution is to provide the appropriate QoS level for multicast application requirements using the Integrated Services (IntServ) [BC+94] or Differentiated services (DiffServ) [BB+98] architectures.

These architectures have been proposed and designed by Internet Engineering Task Force (IETF) [IETF] to provide QoS for IP [JN98] networks. IntServ was the first approach to solve the QoS problem. However the processing of overhead and complexity of this architecture are too heavy for Internet environment and therefore something simpler and more scalable is needed.

The scalability concern becomes the main principle of the DiffServ architecture. However there are disadvantages to DiffServ, such as its static nature, which leads IETF to propose the interoperation of both architectures [BF+00]. The advantage of this interoperation is to provide a dynamic nature, efficient network resources (IntServ) and scalable architecture (DiffServ), which would be able to offer end-to-end services for multicast application requirements.

However each of these individual solutions have several drawbacks. To overcome these drawbacks, the network needs a robust, reliable and scalable multicast transport protocol to work with IntServ or/and DiffServ principles to provide the end-to-end services for multicast application requirements.

1.1 OBJECTIVE AND CONTRIBUTION

The objective of this research is to develop a multicast transport protocol with tree-based and QoS management functionality to provide reliably, scalability, end-to-end resources and dynamic admission control for multicast applications. Enhance Communication Transport Protocol (ECTP), is an experimental tree-based multicast transport protocol specified by ITU (International telecommunication union) [ITU] and ISO (International standard organisation) [ISO], designed to support Internet multicast applications running over multicast capable networks. ECTP transport protocol operates over IPv4/IPv6 [Durand01] networks that have the IP multicast forwarding capability with the help of IGMP [Fenner97] and IP multicast routing protocols [Moy97]. Figure (1.1) shows the ECTP transport protocol inside the TCP/IP model.

Figure (1.1): ECTP transport protocol inside TCP/IP model

ECTP specifications have been defined by ITU.T/ISO in two parts: (i) Control and (ii) QoS management for simplex connections. The ECTP control specification, which has already been completed [Koh01] specifies the protocol operations for connection establishment and tree-based error control to provide reliability and scalability for multicast applications.

This research addresses the further enhancement and implementation of ECTP QoS management specification [Koh02]. It describes QoS management functions (negotiation, monitoring and maintenance) for the stable management of QoS for users in a simplex connection. QoS management enables the ECTP transport protocol to negotiate QoS management functionality provided by IntServ or/and DiffServ principals in networks. Therefore, ECTP transport protocol is able to provide end-to-end resources, dynamic admission control and scalability on networks.

This research proposes two models to enable ECTP transport protocol with QoS management functionality, which are described as follow:

MODEL (I)- this enables ECTP transport protocol to negotiate end-to-end resource reservation using the standard RSVP [BZ+97] signalling protocol. This model proposes to map ECTP QoS parameters target values into standard RSVP signalling protocol to negotiate end-to-end quality of service over the network.

This mapping enables the ECTP transport protocol to provide end-to-end connection for multicast applications by negotiating the QoS level in connection creation phase. QoS negotiation operation uses RSVP signalling to enable ECTP to negotiate the resource reservation required by applications to all active receivers before transmitting packets. The ECTP application programming interface (ECTP API) has been designed by ITU.T/ISO to interact with multicast applications that utilize ECTP transport capabilities. The RSVP application programming interface (RAPI) has been designed by IETF to interact with the multicast applications in order to request required QoS. Figure (1.2) illustrates the interworking of ECTP with standard RSVP.

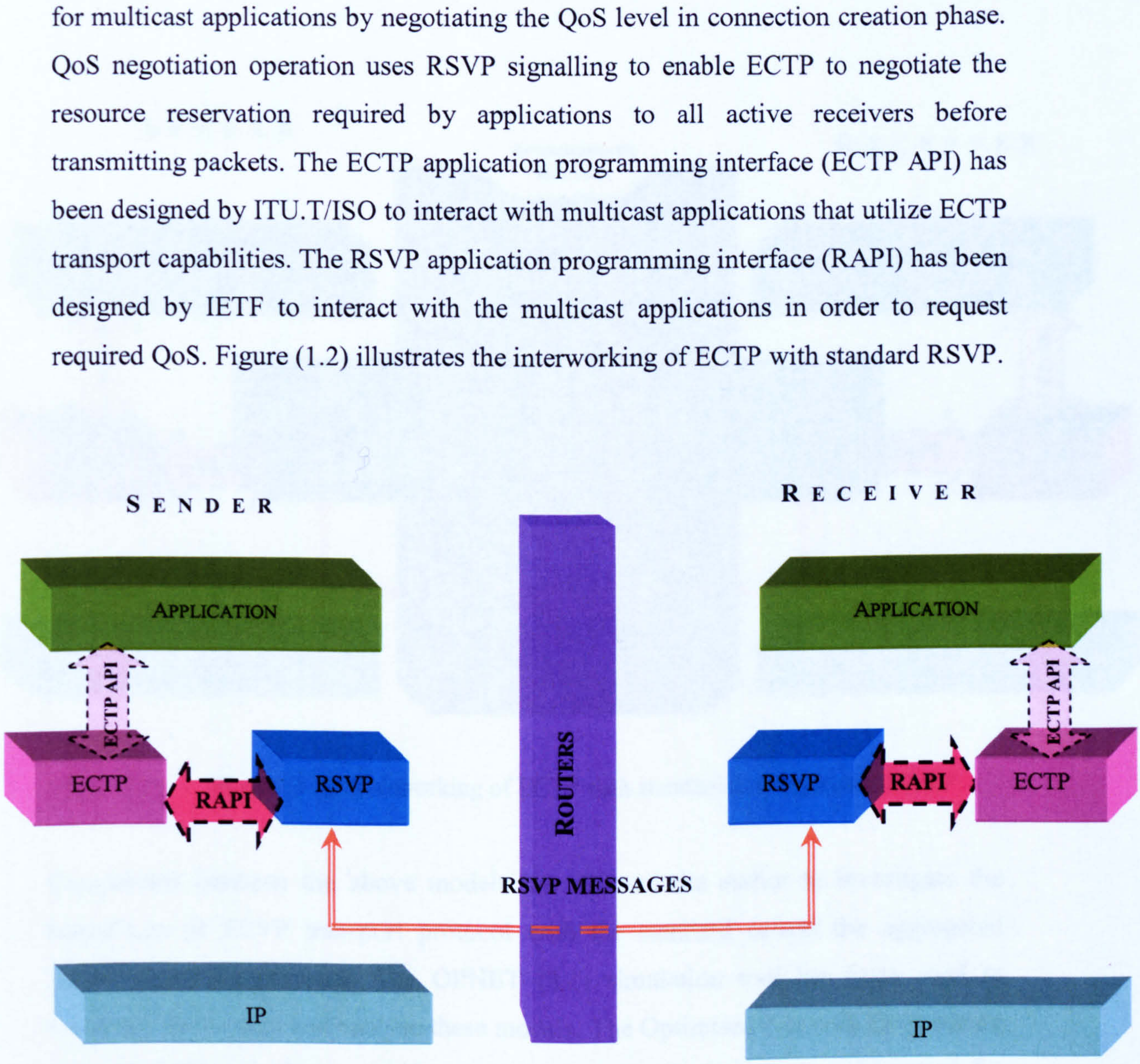


Figure (1.2): Interworking of ECTP with standard RSVP

MODEL (II)- This enables the ECTP transport protocol to negotiate end-to-end resource reservation using the standard and aggregated RSVP signalling protocol.
The pervious model only uses IntServ principal; therefore it cannot provide any scalability over network. This model proposes to map ECTP QoS parameters target values into standard and aggregated RSVP object traffic descriptors. Therefore, this model provides end-to-end resources and dynamic admission control as well as scalability over the network. Figure (1.3) illustrates interworking between ECTP with standard and aggregated RSVP.

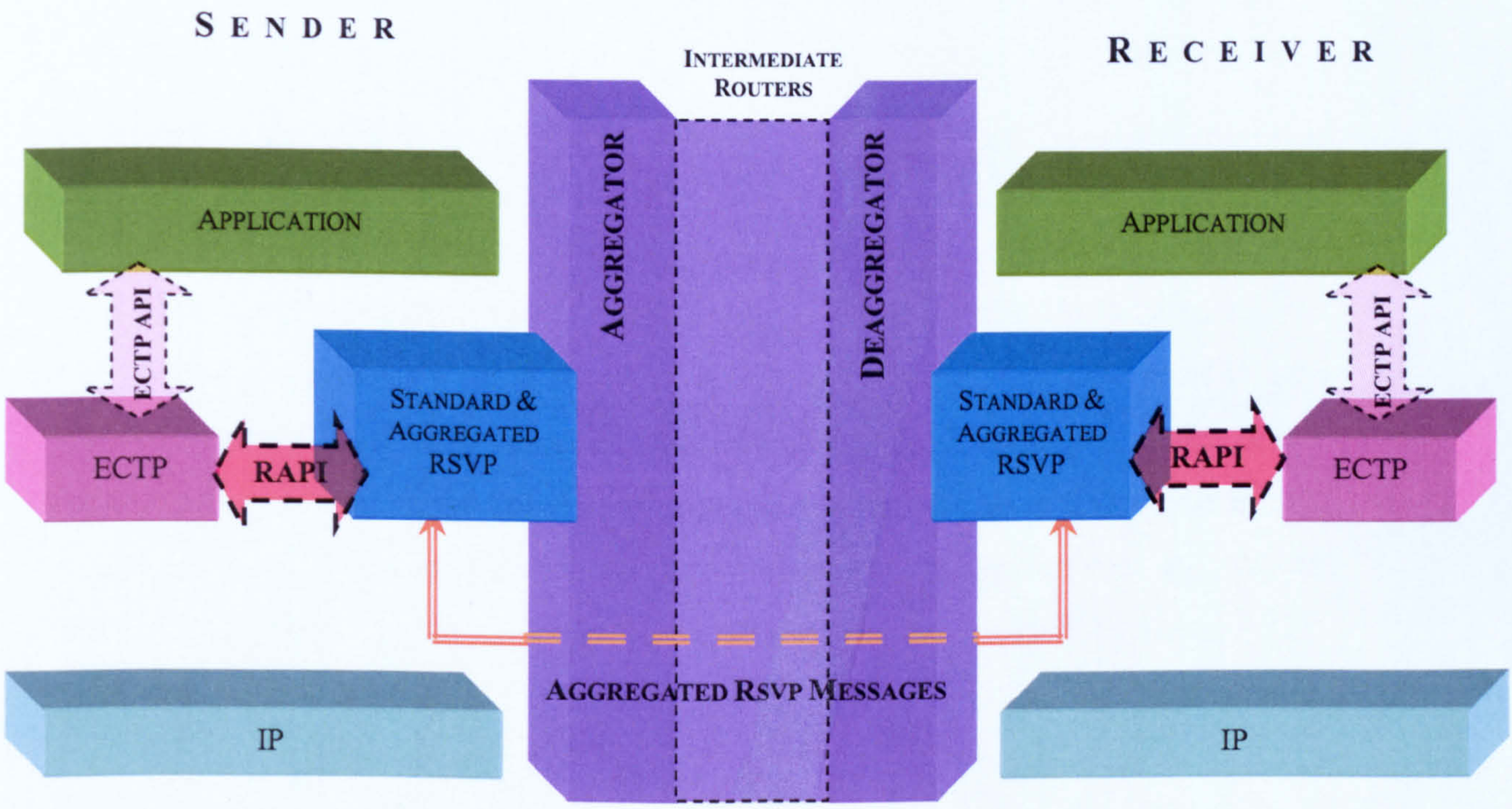


Figure (1.3): Interworking of ECTP with standard and aggregated RSVP

Comparison between the above models has allowed the author to investigate the behaviours of ECTP transport protocol with the standard or/and the aggregated RSVP signalling protocol. The OPNET (8.1) simulation tool has been used to construct, implement and analyse these models. The Optimised Network Engineering Tool (OPNET) simulator provides a comprehensive development environment for modelling and performance of communications networks. It was employed in order to achieve the following results and conclusions:

- To show that enabling the ECTP transport protocol with QoS management provides more efficient end-to-end services for multicast applications compared to the UDP transport protocol.

- To show that enabling the ECTP transport protocol with the standard and aggregated RSVP (model (II)) provides more efficient end-to-end services compared to the ECTP with the standard RSVP (Model (I)) for multicast applications.

CHAPTER 2

MULTICAST SERVICES

1.2 THESIS OUTLINE

The rest of this thesis is arranged as follows: *CHAPTER (2) MULTICAST SERVICES* reviews the multicast service behaviour, the multicast applications (types and categories and their requirements) and discusses the multicast transport protocols (their components and approaches). It also describes in detail the first part of ECTP control specification.

CHAPTER (3) QUALITY OF SERVICE reviews the Quality of Service (QoS) terminology and behaviours. It discusses the QoS management (its parameters and components). It describes the QoS services classification: Integrated Services (IntServ) and Differentiated Services (DiffServ) and reviews the queuing mechanisms. It describes the design of interoperation between IntServ and DiffServ, and their proposed reference network model.

CHAPTER (4) QOS MANAGEMENT FUNCTIONALITY IN ECTP TRANSPORT PROTOCOL describes the second part of ECTP specifications, which enables QoS management functionality into ECTP transport protocol. It reviews the overall ECTP lifetime connections with the corresponding QoS management operations, defines three target values for the QoS parameters and describes the behaviour of the ECTP QoS management operations: Negotiation, Monitoring and Maintenance throughout the ECTP connection.

CHAPTER (5) PROPOSED MODELS AND ARCHITECTURES describes the proposed models and their architectures. It discusses the proposed models: Model (I) and Model (II) and reviews the interworking between ECTP QoS parameters and RSVP object traffic descriptors. It describes the designs and architectures of Model (I)- "Enabling ECTP protocol to negotiate end-to-end resource reservation using the standard RSVP" and the Model (II)- "Enabling ECTP protocol to negotiate end-to-end resource reservation using the standard and aggregated RSVP".

CHAPTER (6) DESIGN AND IMPLEMENTATIONS presents the OPNET modelling methodology, its functions and discusses the construction of OPNET modules and their corresponding state transition diagrams (STD) for each node.

CHAPTER 7- SIMULATION RESULTS AND ANALYSES presents the experimental results of three case studies and analyses them in OPNET for the proposed models. Finally, *CHAPTER 8- CONCLUSION AND FUTURE WORK* concludes this thesis and discusses the possibility for future work.

2. MULTICAST SERVICES

2.1 TERMINOLOGY

There are an increasing number of applications such as software distribution, audio/video conferences, and audio/video broadcasts where data is destined to multiple receivers. Multicasting service is defined as sending data from one source to a group of destinations. The motivation for developing multicast is that there are applications for which a packet needs to be sent to more than one destination. Instead of forcing the source host to send a separate packet to each destination, the source would be able to send a single packet to multicast addresses, and for the network to deliver a copy of that packet to each group of hosts. Hosts can then choose to join or leave this group without synchronizing with other members. A host may also belong to more than one group at a time.

This approach contrasts with unicast or broadcast services. Unicast or broadcast services has been defined as sending data from one source to all destinations, whereas multicast sends the data to only those who wants their receivers receive the information.

Multicast service is a receivers-based concept which receivers join a particular multicast session group and traffic is delivered to all members of that group by the network infrastructure. The sender does not need to maintain a list of receivers. Only one copy of each multicast message will pass over any link in the network, and copies of the message will be made only where paths diverge at a router. [Mir01]

2.2 UNICAST VS MULTICAST

The exact definition for unicast can be described as sending a packet from one sender to one receiver. "Transmission Control Protocol" (TCP) [Postel81] is, by its own nature, unicast oriented whereas "User Datagram Protocol" (UDP) [Postel80] supports a lot more

paradigms, but if sender is sending UDP packets and there is only one process supposed to receive them, this is unicast too.

For years unicast transmissions proved to be enough for Internet until the first implementation of multicast applications emerged (section (2.4)). [Miller99]

With today's technology it is possible to afford the "cost" of making a unicast connection with everyone who wants to see a web page. However, multicast applications need a huge amount of bandwidth compared with web applications. Before multicast came into scene there were two options for unicast connections [PA+01]:

1) Establishment of a separate unicast connection with each of the recipients,
   A single connection sending audio/video consumes a huge bandwidth, imagine having to establish hundreds or, may be thousands of those connections. Both the sending computer and the network would collapse.

2) Broadcast
   Broadcast seems to be *a* solution, but it's not certainly *the* solution. If all the hosts in LAN attend the conference, it may use broadcast. Packets will be sent only once and every host will receive them as they are sent to the broadcast address. The problem is that perhaps only a *few* of the hosts and not *all* are interested in those packets. Furthermore: some hosts may be interested in the conference, but are outside of the LAN, a few routers away. Thus broadcast only works fine inside a LAN, and can not be routed across different LANs.

The best solution seems to be one in which packets send to a certain special address, (multicast). Then, all hosts, which have decided to join the conference, will be aware of packets with that destination address read them when they traverse the network, and pass them to the IP layer to be demultiplexed.

This is similar to broadcasting, which a sender only sends one broadcast packet and all the hosts in the network recognize and read it; it differs, however, in that not all multicast packets are read and processed, but only those that were previously registered. [Comer91]

2.3 MULTICAST ENABLED NETWORK

A " multicast-enable network" provides end-to-end services in the network infrastructure to allow any host to send packets to a multicast address [Almeroth99]. There are two kinds of multicast-enable networks available, which are described as follow:

- This model is known by a number of names including "Internet Standard Multicast (ISM)", " Internet Traditional Multicast (ITM)" or "Deering Multicast" [Deering89], where a receiver simply joins the group and does not need to know the identity of the source (s).

- This model is called " Source Specific Multicast (SSM)",[HC01] which modifies the original service model such that in addition to knowing the group address, a receiver must know the set of relevant sources. Currently, the only way for an application to know the type of multicast is based on the SSM group address.

2.4 REVIEW OF MULTICAST APPLICATIONS

A "multicast application" is defined as any application that sends to and/or receives from IP multicast address. These applications are poised to become important enablers over both private networks and Internet. In today's Internet, multicast applications are categorized into three types and four categories, depending on the number of actively transmitting source that they are able to manage. [QA01]

2.4.1 MULTICAST APPLICATIONS TYPES

- **ONE-TO-MANY APPLICATIONS (1TOM)**

  One-to-many applications have a single sender, and multiple simultaneous receivers. The large majority of developers have seen one-to-many multicast as a tool to deliver high-bandwidth integrated audio and video streams directly to

individual network users. Instead of flooding a network with many broadcast data, the provider can use multicast to send the application. [Estrin98]

- **MANY-TO-MANY APPLICATIONS (MTOM)**

  In many-to-many applications two or more of the receivers also act as senders. In other words, MtoM applications are characterized by two-way multicast communications. [PMB99]

- **MANY-TO-ONE APPLICATIONS (MTO1)**

  Many-to-one applications involve "two-way" communication between a potentially rather large set of source and single destination. Mto1 application is an artificial category defined by the application-level relationship between sender (s) and receiver. [HNRS99]

2.4.2 MULTICAST APPLICATION CATEGORIES

- **MULTIMEDIA REAL-TIME APPLICATIONS**

  These applications come in two flavors: a conferencing many-to-many application set and a one-to-many event application set. In the conferencing application set, each member of group is both a receiver and transmitter. The second real time multimedia application set is created as an event, similar to broadcast television. Real-time multimedia applications do not require absolute reliability but rather tight limits on timing jitter. Data needs to be delivered in a smooth flow to ensure that motion is captured properly and looks natural, and that lip synchronization is correct. [Bufrod94]

- **MULTIMEDIA NON-REAL-TIME APPLICATIONS**

  Multimedia one-to-many streaming applications, such as remote classrooms and events, may wish to be transmitted in non-real-time manner for reply later. The reasons are a) the information does not want to view when it is delivered, b) the network cannot support high-quality multimedia at the time of the event. Another

large category of non-real-time application includes the mirroring (replication) and caching of web sites to bring content closer to the user. [Bufrod94]

- **DATA REAL-TIME (STREAMING) APPLICATION**

  These applications similarly split into two basic types: Conferencing/ distributed games and event streams.

  A variety of one-to-many data streaming applications has emerged. The one-to-many data streaming are financial data, stock, commodity and other quotes for financial products. For the casual user who uses these streams, the data need not be absolutely reliable.

  A second one-to-many data streaming application involves news feeds. Data is sent out as stream, usually in text form. This stream has a reliability requirement but does not have the stringent latency and concurrency requirements of some data feed applications. Scaling requirements can be high. [Bufrod94]

- **DATA NON-REAL TIME APPLICATIONS**

  Virtually all of the many non-real-time data applications require reliability. One of the most compelling is the distribution of software to remote sites from a central headquarters. Another major application category is called push applications; these applications do not scale, and will be widely deployed only if they embrace reliable multicast technology for distribution. Also other large applications of non-real-time multimedia data distribution include the mirroring and caching of web sites to bring content closer to users. [Bufrod94]

Table (2.1) summarizes the multicast applications types and categories. Appendix A has explained details of types of Multicast applications.

CATEGORIES / CLASSIFICATIONS	ONE-TO-MANY (1TOM)	MANY-TO-MANY (MTOM)	MANY-TO-ONE (MTO1)
MULTIMEDIA REAL -TIME	Multimedia events Scheduled audio/video distribution	Distance learning Distribution Interactive simulations (DIS) Jam sessions	
MULTIMEDIA NON REAL-TIME	Replication File Distribution and Caching	Chat groups	Jukebox
DATA-ONLY REAL-TIME	Push media News feeds Monitoring Stock quotes White boarding	Multimedia Conferencing Collaboration	Data Collection Auctions Polling
DATA-ONLY NON REAL-TIME		Synchronized Resources Database replication Concurrent processing Multi-player games	Resource Discovery Accounting

Table (2.1)- Multicast applications types and categories

2.4.3 MULTICAST APPLICATION REQUIREMENTS

Multicast applications have varying performance requirements; therefore to design a service that serves quality of service satisfaction (guaranteed), it is important to determine the requirements that the applications are interested in and the desired quality of service level for each requirement. [LB01]

There exist two types of requirements for multicast applications known as *common* and *unique* requirements. This work is based on the common requirements of multicast applications. The latter requirements are beyond of scope of this work. [QA01]

2.4.3.1 COMMON REQUIREMENTS

- **THROUGHPUT OR BANDWIDTH RATE**

  Throughput has been considered to be one of the most important QoS parameters in data communication. This is reasonable because throughput directly reflects how much information the network is able to deliver during a certain time interval. In other words, it represents an amount of application

data over a specific time period. [QA01] Table (2.2) summarizes all types of multicast applications on the scale of sensitivity to throughput. [Miller99]

	LOW THROUGHPUT	MEDIAN THROUGHPUT	HIGH THROUGHPUT
1TOM	Push media Announcements	File distribution and caching Monitoring	Scheduled audio/video
MTOM	Chat groups	Synchronized resources Distributed Interactive simulation	Multimedia conferencing Concurrent Processing Distance learning Multi-player games Jam sessions
MTO1	Resource discovery Auctions Polling	Data collection Accounting	Jukebox

Table (2.2): Types of multicast applications on the scale of sensitivity to throughput

- **TRANSIT DELAY**

  All multicast applications are delay-sensitive because the transmitted information needs to be re-played at the receiver end in real-time. For example, the delay of live video transmission has a close relationship to the receiver's latest play back time.

  It is also conceivable that delay varies from one application to another due to the differences in their delay-sensitivities and play back environment set-ups. For example, a non-interactive real-time video transmission may tolerate larger network delay than an interactive application [QA01]. Table (2.3) shows all types of multicast applications on the scale of sensitivity to deliver delays. [Miller99]

	DELAY TOLERANT	MEDIAN DELAY	DELAY INTOLERANT
1TOM	Push media File distribution and caching	Scheduled audio/video Announcements	Monitoring
MTOM		Synchronized resources Collaboration Distance learning Chat groups	Multimedia conferencing Concurrent Processing Distributed Interactive simulation Multi-player games Jam sessions
MTO1	Polling	Resource discovery Data collection Jukebox Accounting	Auctions

Table (2.3): Types of multicast applications on the scale of sensitivity to deliver delays

- **TRANSIT DELAY JITTER**

  Jitter is defined as a variation in the delay of received packets. At the sending side, packets are sent in a continuous stream with the packets spaced evenly apart. Due to network congestion, improper queuing, or configuration errors, this steady stream can become lumpy, or the delay between each packet can vary instead of remaining constant. [QA01]

- **DATA LOSS**

  When network congestion occurs, packets may be dropped due to an over-flowing buffer or because they arrive too late to be used. Those dropped packets directly affect the viewing quality at the receiving end. The data loss rate is defined as the ratio of the amount of lost data to the amount of transmitted data. [QA01]

2.4.3.2 UNIQUE REQUIREMENTS

Unlike one-to-many applications, which just need the common requirements, the many-to-many and many-to-one applications require the following unique requirements (which are out of scope of this work). Table (2.4) summarizes the unique requirements of multicast applications. [QA01], [Maufer98]

UNIQUE REQUIREMENTS	
Address management	Selection and coordinated of address allocation
Session management	Perform application-layer services on top of multicast transport.
Heterogeneous receiver support	Sending to receivers with a wide variety of common requirements.
Reliable data delivery	Ensuring that all data sent is received by all receivers.
Security	Ensuring content privacy among dynamic multicast group memberships, and limiting senders
Synchronized play out	Allow multiple receives to replay data received in synchronized fashion.

Table (2.4): Multicast applications unique requirements

2.5 MULTICAST ROUTING PROTOCOLS

As it mentioned in section (2.2), Multicast service is distinct from repetitive unicast. Multicast establishes for routing a tree structure that connects source with the receivers. The multicast tree is rooted at the sender and leaves are receivers. Multicast delivery sends data across this tree toward the receivers. As opposed to unicast delivery, data not copied at the source, but is copied inside the network at branch points of the multicast distribution tree. The exciting multicast routing protocols are: Distance-Vector Multicast Routing Protocol (DVMRP), core-base trees (CBT), protocol-independent multicast (PIM) and the multicast Internet protocol (MIP). [Moy97]

2.6 MULTICAST TRANSPORT PROTOCOLS

2.6.1 COMPONENTS

As it has mentioned in section (2.4), different multicast applications have different requirements, which constrain the design space in ways that two applications with differing requirements often cannot share a single solution. There are however many successful reliable multicast protocol designs that satisfactory serve desired requirements. [HF+00]

Three components have been defined that each multicast transport protocol must have in its own structure [MR+98], which are as follow:

- **SCALABILITY**

  The protocol should be able to work under a variety of conditions that includes multiple network topologies, link speeds and the receiver set size. The protocol should use scalable group membership algorithms, where membership can either be anonymous (the sender does not know the list of receivers), or fully distributed (sender receives a count of the number of receivers).

- **RELIABILITY**

  The protocol must be safe to deploy in the widespread of Internet and should be reliable to provide best service level for multicast applications throughout the network.

- **SECURITY**

  The protocol must be analyzed to show what is necessary to allow it to cope with security and privacy issues. This includes understanding the role of protocol in data confidentiality and sender authentication, as well as how the protocol will provide defenses against denial of service attacks. Security is out of scope of this work.

2.6.2 Scalability

The existing multicast protocols have some of the above components, especially the scalability. Multicast transport protocol should be scalable to work under a variety of conditions that include multiple network topologies. There are four generic approaches for scalability, which each multicast protocol should be mapped into each class. [MB+99] Each multicast transport protocol assumes the existence of IP multicast routing tree provided by underlying IP multicast routing protocols such as DVMRP, CBT, PIM and MIP. [Moy97]

- **SENDER-INITIATED APPROACH**

  A sender-initiated approach requires the sender to maintain a list (called ACK list) for each packet, of the receivers from which it has received a positive acknowledgement (ACK) [MB+99]. Each times a receiver correctly receivers a packet, it returns an ACK. Upon receipt of ACK, the sender updates the ACK list for the corresponding packet. Lost packets are dealt with throughout the use of times. Sender starts a timer at the time of packet transmission and, if it expires prior to the sender having

receivers ACKs for this packet from every receiver. The sender retransmits the packet and restarts timer. Receivers are not restricted from contacting the source. It is clear that the source is required to know the constituency of the receiver set, and that the scheme suffers from the ACK implosion problem. Figure (2.1) shows the structure of sender-initiated protocol. [PT+94]

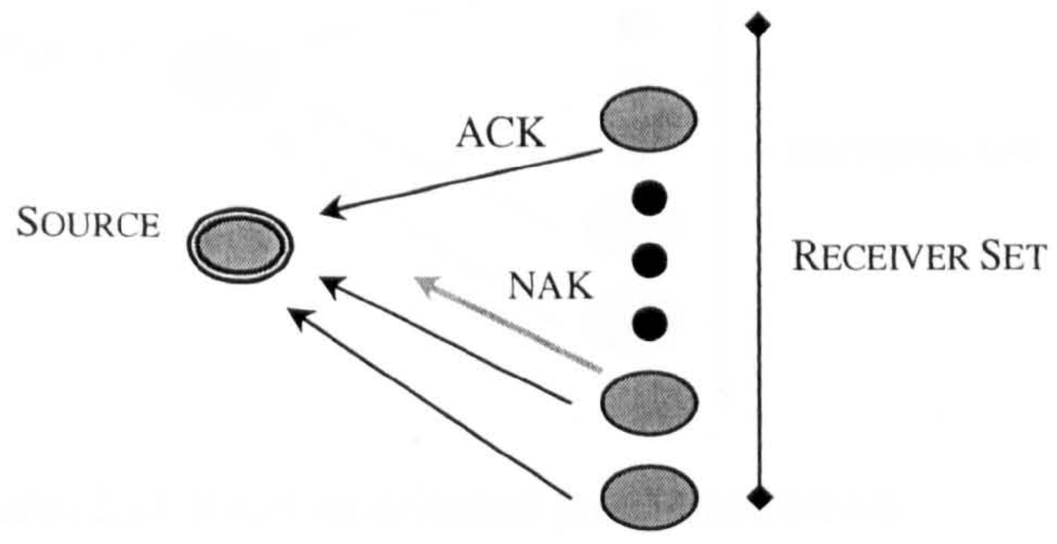


Figure (2.1): Sender-initiated protocol structure

Table (2.5) summarizes the exciting multicast transport protocols, which has the structure of sender-initiated approach. [Obraczka98]

MULTICAST TRANSPORT PROTOCOL	DESCRIPTIONS
Xpress transport protocol (XPT)	Support a wide range of applications and thus provide all the functionality of TCP, UDP.
Reliable broadcast protocol (RBP)	Provides point communication between sites connected by a local-area broadcast network.
Multicast dissemination protocol (MDP)	Provides a reliable multicast framework for file distribution.

Table (2.5): list of multicast transport protocols with the sender-initiated approach

- **RECEIVER-INITIATED APPROACH**

A receiver-initiated approach places the responsibility for ensuring reliable packet delivery on the receiver. The sender continues to transmit new data packets until it receives a negative acknowledgment (NAK) from a receiver [MB+99]. When this occurs, the sender then retransmits the packet required by that receiver. The role of receiver is to check for lost packets. If it decides that it has not received a

particular packet, it transmits a NAK to the sender. In order to guard either the loss of the NAK or the subsequent packet retransmission, the receiver uses timers in a manner similar to sender-initiated protocols. Figure (2.2) shows the structure of receiver -initiated protocol. [PT+94]

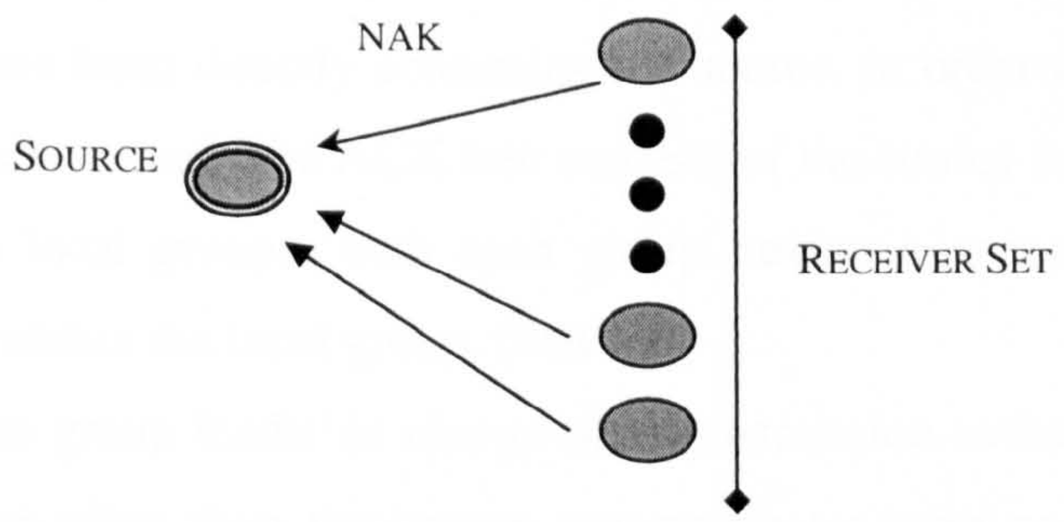


Figure (2.2): Receiver-initiated protocol structure

- **RECEIVER-INITIATED WITH NAK-AVOIDANCE**

Because receivers communicate NAKs back to the source, receiver-initiated approach have the possibility of experiencing an NAK-implosion problem at the source if many receivers detect transmission errors. To remedy this problem, receiver-initiated protocols adopt the NAK-avoidance scheme.

The sender multicasts all packets and state information, giving priority to retransmission. Whenever a receiver detects a packet loss, it waits for a random time period and then multicasts a NAK to the sender and all other receivers. [RJ87]

Table (2.6) summarizes the exciting multicast transport protocols, which has the structure of Receiver-initiated approach. [Obraczka98]

MULTICAST TRANSPORT PROTOCOL	DESCRIPTIONS
Scalable reliable multicast (SRM)	Provide reliability for distributed data.
Log-based receiver-reliable multicast (LBRM)	Loges all packets transmitted by the source and retransmitted lost packets.
Transport protocol for reliable multicast (TRM)	Provides reliable multipoint communication to support interactive multimedia applications.
Real-time transport protocol (RTP)	Support multiparty multimedia conferencing services over Internet.

Table (2.6): List of multicast transport protocols with the Receiver-initiated approach

- **TREE-BASED APPROACH**

Tree-based approach is characterized by dividing the receiver set into groups, distributing retransmission responsibility over an acknowledgement tree (ACK tree) structure built from the set of groups, with the source as the root of the tree. A simple illustration of a tree-based protocol is presented in figure (2.3). The ACK tree structure prevents receivers from directly contacting the source, in order to maintain scalability with a large receiver set. The ACK tree consists of the source (sender) and receivers organized into local groups, with each group having a group leader in charge of retransmission within the local group. [MB+99]

The source (sender) is the group leader in charge of retransmission to its own local group. Each group leader other than the source communicates with another local group (to either a child or the group leader) closer to the source to request retransmissions of packets that are not received correctly. Group leaders may be children of another local group, or minimally may just be in contact with another local group. Each local group may have more than one group leader to handle multiple sources.

Hosts that are only children are at the bottom of the ACK tree, and are termed "leaves". Obviously, an ACK tree consisting of the source as the only leader and leaf nodes corresponds to the *sender-initiated* scheme. [JH+99]

Acknowledgements from children in a group, including the source's own group, are sent only to the group leader. The children of a group send their acknowledgements to the group leader as soon as they receive correct packets. These acknowledgements refer as local ACKs or local NAKs, i.e. retransmissions are trigged by local ACKs and local NAKs unicast to group leaders by their children. Similar to sender-initiated schemes, the use of local NAKs is unnecessary for correct operation of the protocol. [LL+96]

Tree-based protocols can also delegate to leaders of sub-trees the decision of when to delete packets, which is conditional upon receipt of aggregate ACKs from the children of the group. Aggregate ACKs start from the leaves of the ACK tree, and propagate toward the source, one local group at a time. A group leader cannot send an

aggregate ACK until all its children have sent an aggregate ACK. Using aggregate ACKs is necessary to ensure that the protocol operates correctly even if group leaders fail, or if the ACK tree is partitioned for long periods of time. If aggregate ACKs are not used, i.e. if a group leader only waits for all its children to send local ACKs, then correct operation after group leaders fail can only be guaranteed by not allowing nodes to delete packets.

The use of local ACKs and local NAKs for requesting retransmission is important for throughput. If the source scheduled retransmissions based on aggregate ACKs, it would have to be paced based on the slowest path in the ACK tree. Instead, retransmissions are scheduled independently in each local group.

Tree-based protocols eliminate the ACK-implosion problem, which free the source from having to know the receiver set, and operate solely on message exchanged in local groups (between a group leader and its children in the ACK tree).

Furthermore, if aggregate ACKs are used, a tree-based protocol can work correctly with finite memory even in the presence of receiver failures and network partitions. [LP96]

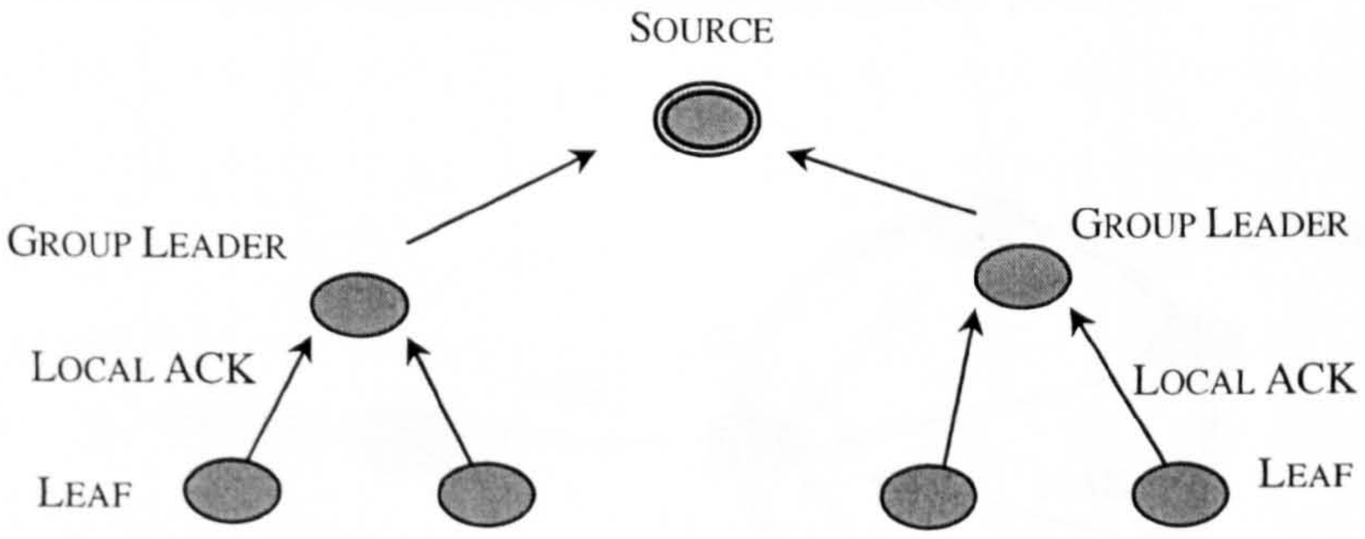


Figure (2.3): Tree-based protocol structure

Table (2.7) shows the exciting multicast transport protocols, which has the structure of Tree-based approach. [Obraczka98]

MULTICAST TRANSPORT PROTOCOL	DESCRIPTIONS
Tree-based multicast transport protocol (TMTP)	Provides reliable multicast communication support for one-t-many bulk data dissemination application.
Reliable multicast transport protocol (RMTP)	Supports reliable one-to-many bulk data delivery
The multicast file transfer protocol (MFTP)	Supports reliable one-to-many data transmission.

Table (2.7): List of multicast transport protocols with the Tree-based approach

- **RING-BASE APPROACH**

The Ring-based approach was originally developed to provide support for applications that require total ordering of transmissions at all receivers [MB+99].. A ring-based protocol is illustrated in figure (2.4). The basic premise is to have only one token site responsible for ACKing packets back to the source. The source times out and retransmits packets it does not receiver an ACK from the token site within a timeout period. Receivers send NAKs to the token site for selective repeat of lost packets that were originally multicast from the source. [CM84]

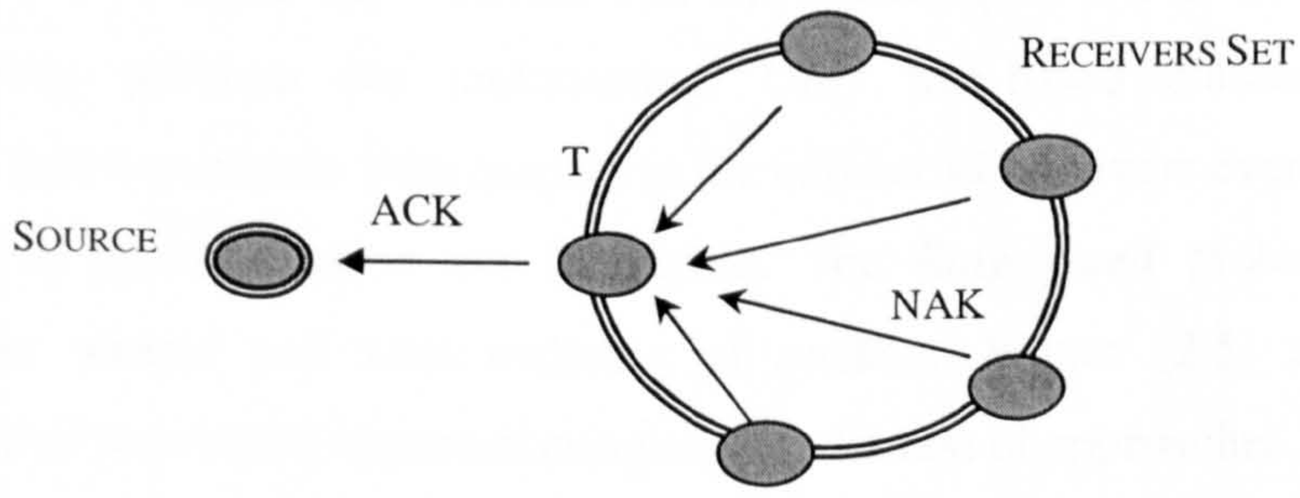


Figure (2.4): Ring-based protocol structure

Table (2.8) shows the exciting multicast transport protocols, which has the structure of ring-base approach. [Obraczka98]

MULTICAST TRANSPORT PROTOCOL	DESCRIPTIONS
Reliable multicast protocol (RMP)	Provides totally ordered, reliable multipoint transport protocol on the top of an unreliable multicast service.
Uniform reliable group communication protocol (URGCP)	Provides reliable and ordered communication among the members of a group.
Multicast Transport Protocol (MTP)	Provides reliable and globally ordered delivery of client data among a group of communicating processes.
Multicast Transport Protocol-2 (MTP-2)	Provides a variant of original MTP, which addresses the requirements of remote collaboration tools such as teleconferencing services.

Table (2.8): List of multicast transport protocols with the Ring-based approach

- **COMPARISONS**

It has been compared and analyzed [LG98] the behavior of the four approaches. The results showed how they are scalable for multicast applications requirements. The *sender-initiated* approach is not scalable at all since the source must account for every receiver listening. The *receiver-initiated* approach is more scalable, especially when NAK-suppression schemes are used to avoid overloading the source with retransmission requests. However, because of the unbounded-memory requirement, this protocol class can only be used efficiently with application-layer support, only for limited applications. The *tree-based* approach is the answer to the scalability problem for multicasting. Only tree-based classes have a throughput that is constants with respect to the number of receivers even when the probability to packet loss is not negligible. The *Ring-based* protocols were designed for atomic and total ordering of packets. Figure (2.5) shows the performance of tree-based approach compared to the rest of approaches. [LVG96]

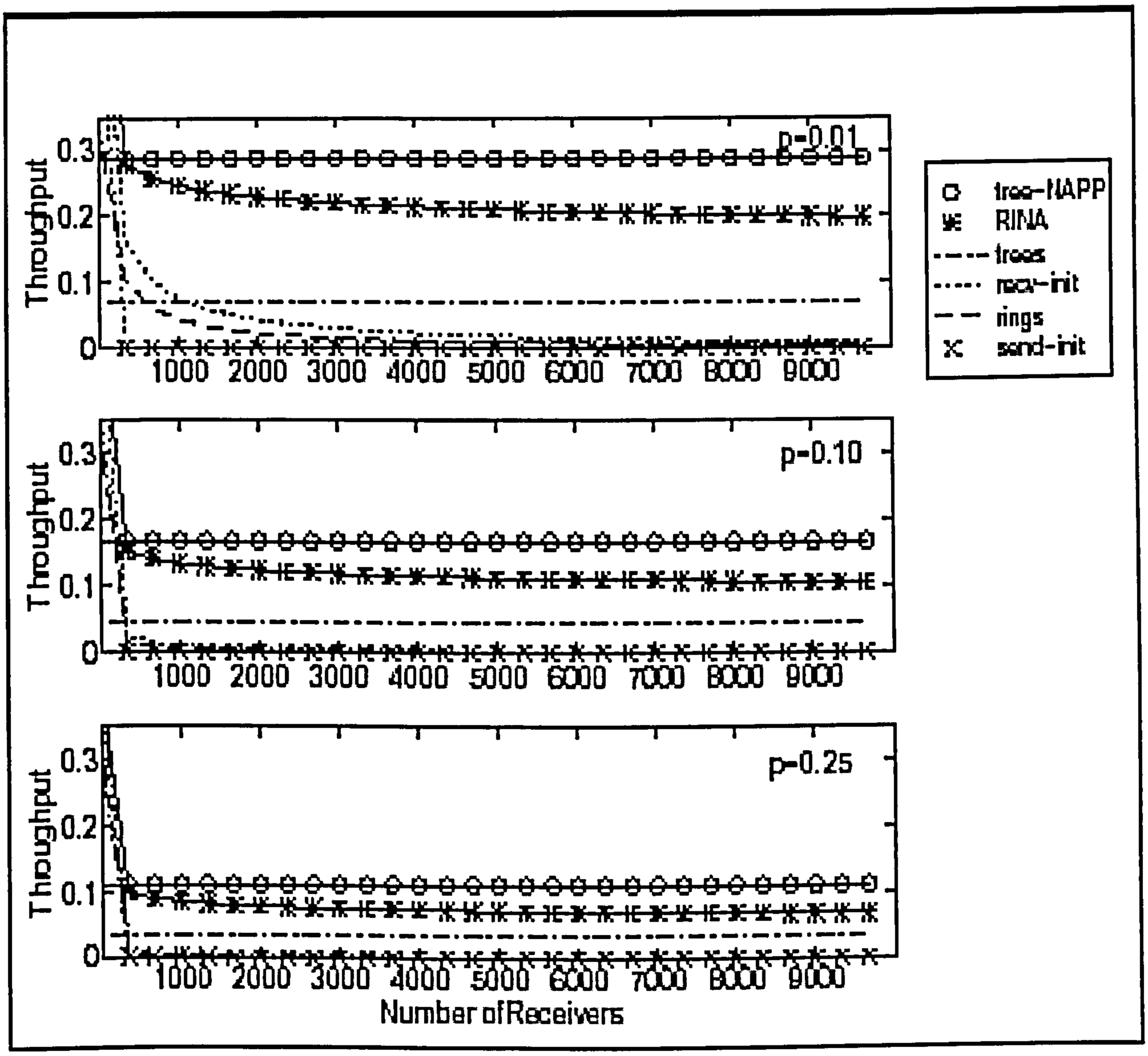


Figure (2.5): Performance of Tree-based approach compared to rest of approaches

2.7 ENHANCED COMMUNICATION TRANSPORT PROTOCOL

It has already been discussed in section (2.6), that a multicast protocol design should be able to provide scalability and reliability to fulfill the multicast applications requirements throughout the network.

This work is based on an experimental tree-based multicast transport protocol called "Enhance Communication Transport Protocol", ECTP. ECTP specifications have been defined by ITU.T (International telecommunication union) [ITU.T] and ISO (International standard organization) [ISO] in two parts: (i) control and (ii) QoS management for simplex connections.

- ECTP-Control specification has already been completed [Koh01] which specifies the protocol operations for connection establishment and scalability by using tree-based approach.
- ECTP-QoS management specification [Koh02], which describes QoS management functions for the stable management of QoS for users in a one-to-many connection.

2.7.1 ECTP TRANSPORT PROTOCOL-CONTROL SPECIFICATION

This section discusses the first part of ECTP transport protocol specification. Enhanced communication transport protocol (ECTP) is a tree-based multicast transport protocol [KK+02] designed to support Internet multicast applications running over IP multicast [QA01] capable networks. ECTP operates over IPv4 [JN98] /IPv6 [Durand01] networks that have the IP multicast forwarding capability.

ECTP features "tight management of multicast sessions" [KK+02] from the sender's perspective. Tight management in the multicast sessions happens when the sender is at the heart of group communications. The sender can perform overall management of the multicast connection establishment; group membership and Quality of Services (QoS) perceived by end users with the help of Internet Group Management Protocol (IGMP)

[Fenner97] and IP multicast routing protocols [Moy97]. Figure (2.6) illustrates ECTP model.

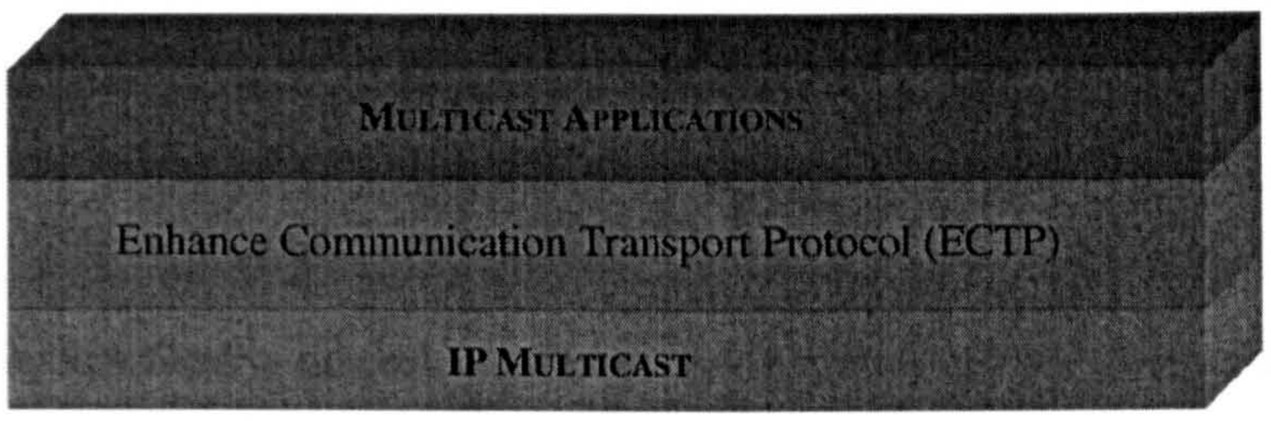


Figure (2.6): ECTP transport protocol inside TCP/IP model

2.7.2 ECTP PROTOCOL DESCRIPTION

In ECTP control specification, the connection management operations have been described in three phases: "*Connection creation phase*", "*Data transmission phase*", "*Connection termination phase*", which are as follow [KK+02]:

- **CONNECTION CREATION PHASE**

The potential receivers are incorporated into the multicast group, called an "enrolled group". Verification processes and group key distribution may be performed together during the enrolment. IP multicast addresses and port numbers must also be announced to the receivers. These enrolment operations may rely on the well-known protocols such as SAP [HP+00]/SDP [HJ98] (Signaling), HTTP (Web Page announcement) and SMTP (E-mail) [Postel82]. This work assumes that the receiver enrolment procedure has been done before the connection creation (Appendix B).

Enrolled receivers can then join/leave the multicast-capable network with the help of the IGMP and IP multicast routing protocols. ECTP is targeted to support tightly controlled multicast connections, i.e. the connection creation and termination functions are explicitly supported, together with QoS negotiation. In addition, connection pause and resumption functions are also provided.

The ECTP sender is designated as the "connection owner", responsible for the overall management of the connection. This node governs the connection creation and termination, connection pause and resumption and late joiners and leavers. The ECTP sender activates the connection creation process by sending a connection creation message. Some or all the enrolled receivers may respond with confirmation messages to the sender. The connection creation is completed when the sender receives the confirmation messages from all of the active receivers. Some or all of the enrolled group receivers will join the connection. The receivers that have joined the connection are called "active receivers".

An enrolled receiver that is *not* active can participate in the connection as a late-joiner. The late-joiner sends a join request to the sender, in response to which the sender transmits a join confirm message. This indicates whether the join request is accepted or not. An active receiver can leave the connection by sending a leaving request to the sender.

- **DATA TRANSMISSION PHASE**

The sender begins to transmit multicast data. For data transmission, an application data stream is sequentially segmented and transmitted by means of data packets to the receivers. Receivers deliver the received data packets to the applications in the same order the sender transmitted them.

During the data transmission, if severe network congestion is indicated by the QoS management functions (negotiation, monitoring and maintenance), the sender suspends the multicast data transmission temporarily. During this period, no new data is delivered, while the sender transmits periodic null data messages to indicate that the sender is "alive". After a pre-specified time has elapsed, the sender resumes the multicast data transmission.

- **CONNECTION TERMINATION PHASE**

The connection termination happens when termination messages are sent to all the receivers, after all the multicast data are transmitted. The connection may also terminate due to a fatal protocol error such as a connection failure.

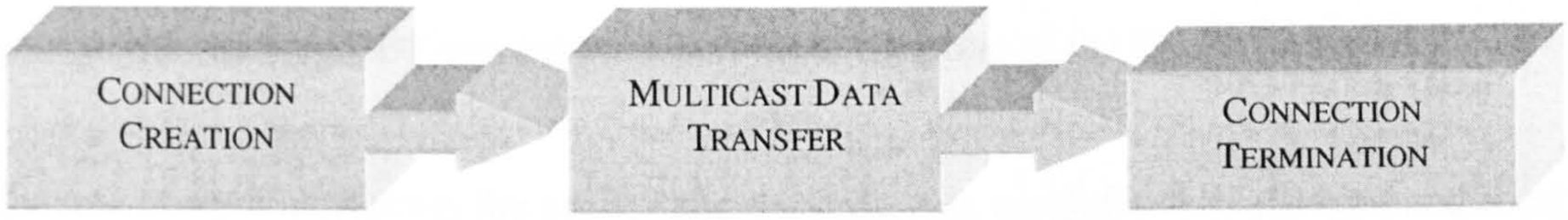


Figure (2.7): ECTP Connection Lifetime

Figure (2.7) shows the ECTP transport connection lifetime. During the transport connection, ECTP provides QoS management functions (chapter 4) for the stable management of the QoS for the connection users. Such QoS management functionality can be achieved with QoS negotiation, monitoring, and maintenance operations.

2.7.3 ECTP TREE-BASED CONTROL MECHANISM

ECTP protocol architecture employs the Tree-based control mechanisms (section (2.6.2) to achieve high scalability as well as reliability by adopting hierarchical tree-based control architecture.

For tree-based reliability control, a hierarchical tree is configured during connection creation. In ECTP tree-based mechanism one multicast group is divided into several local groups, where protocol functionality is performed within the local groups. It provides high scalability because of packet loss, node failure and late joining, which may cause traffic increase. Message overheads are handling locally within the logical tree. In addition, it can reduce the transmission time of error recovery process as the service is provided from a closer node instead of sender. A hierarchical tree is configured during connection. A control tree defines a parent-child relationship between any pair of tree nodes.

ECTP provides the reliability control mechanisms for multicast data transport. ECTP mechanisms are designed to keep congruency with those being proposed in the IETF. To address reliability control with scalability, the IETF has proposed three approaches: Tree based ACK (TRACK) [CW+03], Forward Error Correction (FEC) [LV+02], and Negative ACK Oriented Reliable Multicast (NORM) [AB+04]. Each approach has its own pros and cons, and each service provider may take a different approach toward

implementing reliability control. ECTP adopts the TRACK approach, because it is more similar to the existing TCP mechanisms and more adaptive to the ECTP framework.

In the tree control hierarchy, the sender and the receivers are defined as the root and local groups respectively. A local group consists of a parent and zero or more children. The error, flow and congestion controls are performed for each local group defined by the control tree. The tree configuration currently proposed in IETF could be considered as bottom-up approach, which all the leaf receivers try to find their parent nodes in parallel and the tree is expanded from the leaf receivers towards the root sender. However this approach has some drawbacks, which is inclined to generate more control messages for the tree configuration and a deeper tree with more tree levels than necessary. [KE+01]

ECTP specifies the protocol procedures by using top-down configuration for control tree creation where a control tree is gradually expanded from the sender to the receivers. The Tree- membership is maintained during the connection. A late-joiner is allowed to join the control tree and listens to the heartbeat (HB) messages from one or more on-tree parents and then joins the best parent. [KE+01]

When a child leaves the connection, the parent removers the departing child from the children-list. Node failures are detected by using periodic control messages such as null data (ND), heartbeat (HB) and acknowledgment (ACK). The sender transmits periodic ND messages to indicate that it is alive, even if it has no data to transmit. Each parent sends heartbeat messages to its children. On the other hand, each child transmits periodic ACK messages to its parent.

The error control is performed for each local group defined by a control tree therefore if a child detects a data loss; it sends a retransmission request to its parent via ACK packets. An ACK message contains the information that identifies the data packets, which have been successfully received. Each child can send an ACK message to its parent using one or two ACK generation rules: ACK number and ACK timer. If data traffic is high, an

ACK is generated for the ACK number of data packets. If the traffic is low, an ACK message will be transmitted after the ACK timer expires. [EK+01]

After retransmission of data, the parent activates a retransmission back-off timer. During the time interval, retransmission request (s) for the same data will be ignored. Each parent can remove the data out of its buffer memory, if those have been acknowledged by all of its children.

The flow and congestion control information is delivered from the receiver to the sender, along the control tree. The detailed description of these control procedures is given in QoS management specification (chapter 4). Based on the monitored flow and congestion control information, the sender will adjust the transmission rate.

The sender terminates the connection by sending a termination message to all the receivers, after all the multicast data are transmitted. The connection may also terminate due to a fatal protocol error such as a connection failure. [KE+01]

2.7.4 ECTP CONTROL COMPONENTS

2.7.4.1 NODES

ECTP protocol mechanisms are based on a logical control tree (figure (2.8)), which defines a parent-child relationship between any pair of tree nodes. The following are three node types, which has been classified as follow [KC+01]:

- **TOP OWNER (TO)**

TO is the root (sender) of the control tree in the one-to-many multicast connection. TO manages overall connection management functions, which are as follows:

1. *Connection creation*, in this phase, a control tree is configured by interactions between the sender and the receivers. After the connection is created, TO sends multicast data to the receivers. TO can temporally suspend and resume

the connection. TO can admit or reject the group members who wants to join the existing connection.

2. *Connection termination,* after all the data is transmitted TO terminates the multicast transport connection.

- **LOCAL OWNER (LO)**

  LO is a child (receiver) of the control tree. Each LO has its children that consists of other LOs and/LEs. Each LO retransmits the multicast data that have been lost by its children. It also aggregates the information on the flow and congestion control from its children, and forwards the aggregated information toward TO.

- **LEAF ENTITY (LE)**

  LE is a child (receiver) of the control tree, where each receiver that has not been designated as LO, is referred to as a LE. A LE can not have any children, therefore it located as a leaf node on the control tree.

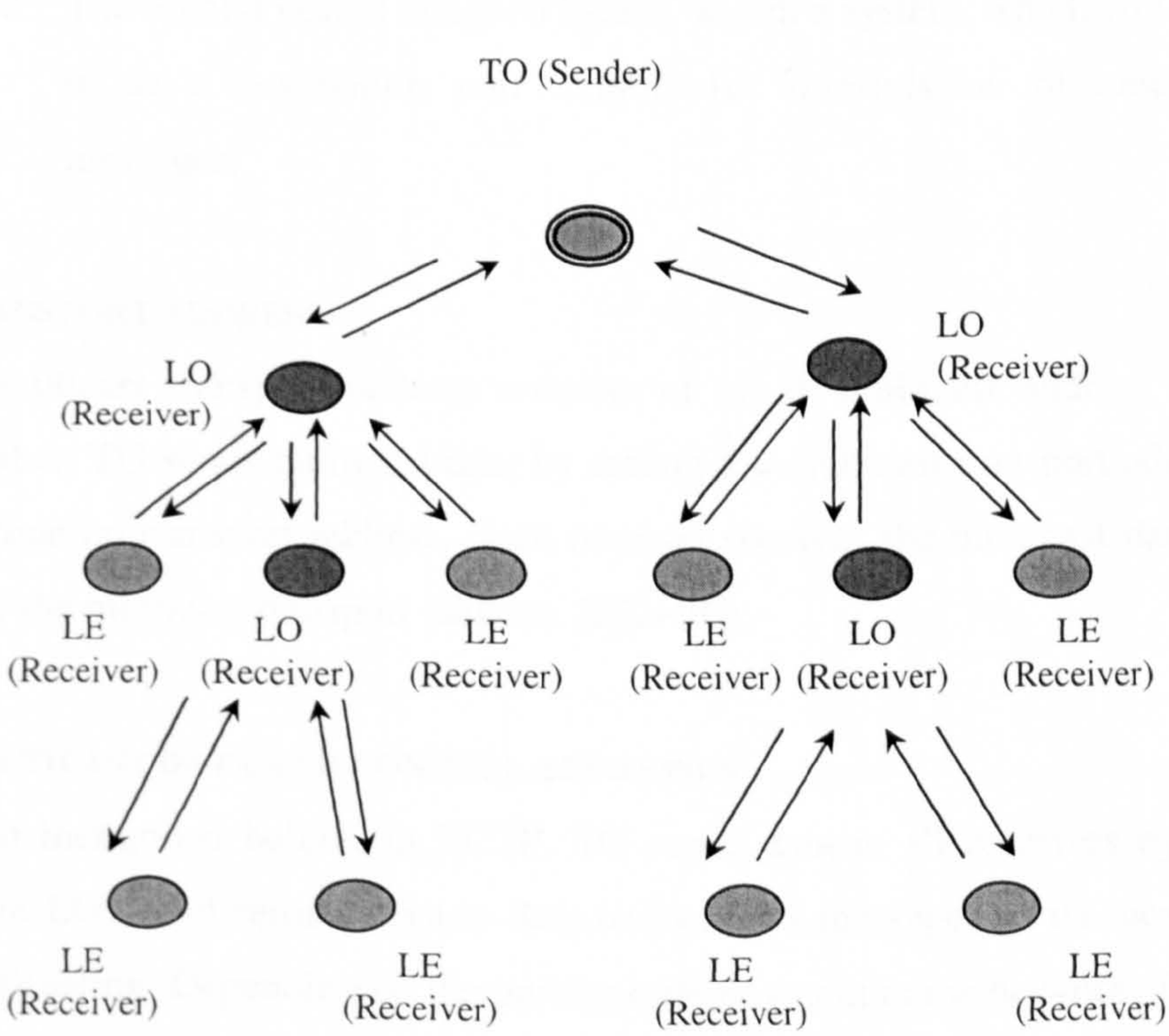


Figure (2.8): ECTP Tree-based Control

After connection is created by TO, TO transmits data to all receivers by multicasting. Each child sends status information on data reception to its parent. The information will then be delivered to TO along the tree control. The multicast data streams flow from TO to LOs and LEs in the downward direction, while the control information is transferred from LEs to TO via LOs in the upward direction along the control tree.

2.7.4.2 ADDRESSING

- **PORT**

  ECTP transport protocol uses a set of ports to identify different applications in IP end host, where the port number is located in each packet header. There are at least two ports that are used in multicast connection, the most common ones are as follows [EK+01]:

  1. The first port is used for multicast data transmission and reception,
  2. The second port is assigned locally within a system, which will be referred to as a destination port number for transmission of unicast control messages.

- **TRANSPORT ADDRESS**

  A multicast transport address consists of an IP multicast address and a port number. TO sends multicast data by setting the multicast transport address as the destination transport address. Each receiver receives the multicast data from TO over the multicast transport address. [EK+01]

- **MULTICAST DATA AND CONTROL ADDRESSES**

  As it mentioned before, in ECTP, TO sends data to all receivers by multicast, while LOs send retransmission data and control messages to its local group by multicasting. Depending on the multicast deployment in the network, TO and LOs may either share a single IP multicast address or use different IP multicast addresses. [EK+01]

2.7.5 PACKET FORMAT

An ECTP packet has two parts: *Fix header* and *Extension elements* or *data parts*. Figure (2.9) shows the ECTP packet format. [PE+01], [Koh01]

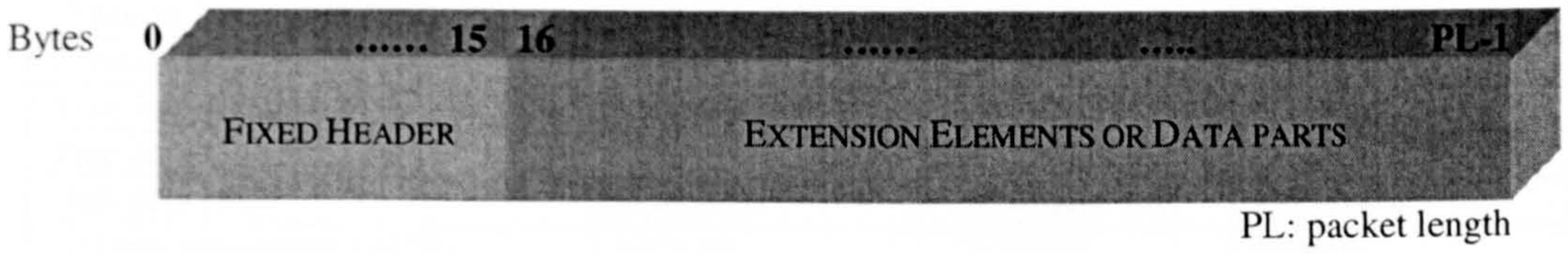


Figure (2.9): ECTP Packet formats

1. FIXED HEADER

The Fixed header consists of 16-byte parameter fields, which is used in all protocol operations and is part of every packet. There are no undefined fields or fields that may at any time have undefined values. Figure (2.10) shows the ECTP fixed header format.

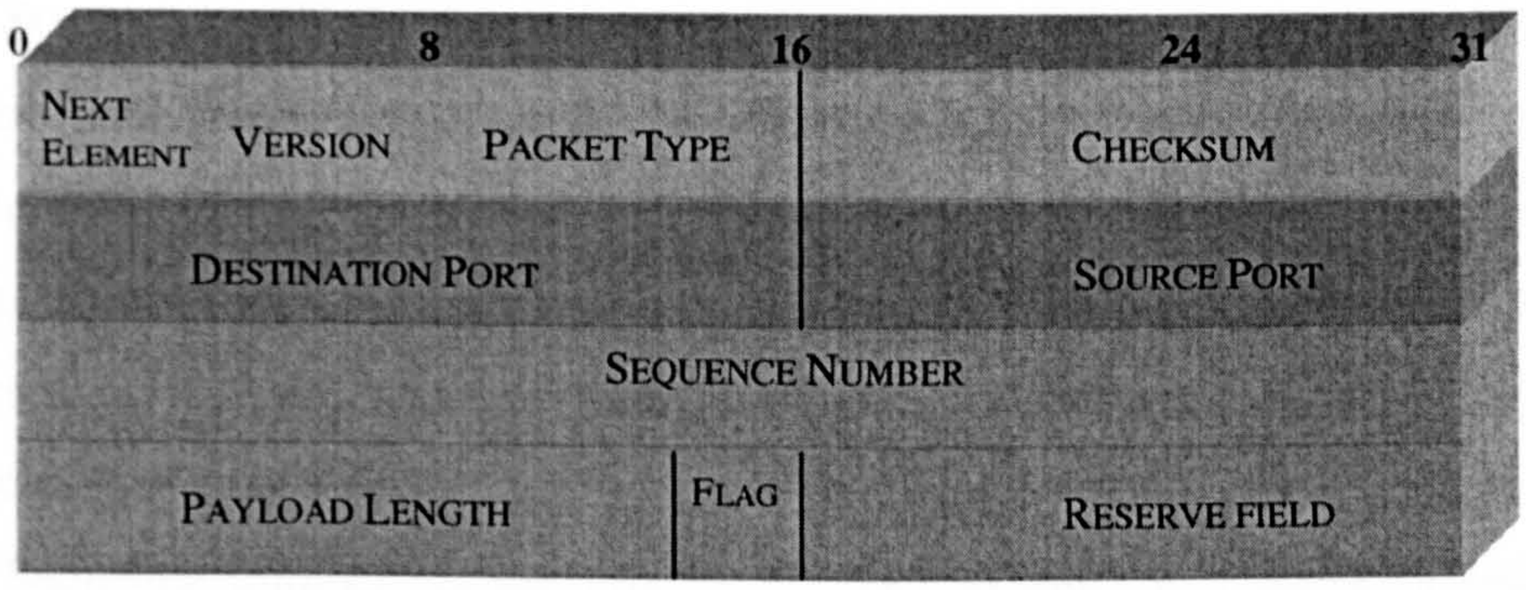


Figure (2.10)- ECTP Fixed header format

The elements in the ECTP fixed header are described as follow:

- **NEXT ELEMENT-** indicates the type of next component immediately following the fixed header.
- **VERSION-** specifies the current version of ECTP protocol.
- **PACKET TYPE-** ECTP packets are classified into two categories, *data packets* and *control packets*. Table (2.9) and (2.10) summarize the packets used in ECTP.

DATA PACKET	TRANSPORT TYPE	FROM	TO
Data (DT)	Multicast	Sender	Receivers
Retransmission Data (RD)	Multicast (Local)	Parent	Children

NOTE1- Sender is TO and Receivers are LOs and LEs.
NOTE2- Parent is TO or LO, and Child is LO or LE

Table (2.9): ECTP Data Packets

CONTROL PACKET	TRANSPORT TYPE	FROM	TO
Creation Request (CR)	Multicast	Sender	Receivers
Creation Confirm (CC)	Unicast	Child	Parent
Tree Join Request (TJ)	Unicast	Child	Parent
Tree Join Confirm (TC)	Unicast	Parent	Child
Null data (ND)	Multicast	Sender	Receivers
Acknowledgement (ACK)	Unicast	Child	Parent
Heartbeat (HB)	Multicast (local)	Parent	Children
Late join Request (JR)	Unicast	Receiver	Sender
Late join Confirm(JC)	Unicast	Sender	Receiver
Leave Request(LR)	Unicast	Parent/child	Child/parent
Connection termination (CT)	Multicast	Sender	Receivers

NOTE1- Sender is TO and Receivers are LOs and LEs.
NOTE2- Parent is TO or LO, and Child is LO or LE.

Table (2.10): ECTP Control Packet

- **SOURCE AND DESTINATION PORTS**

Source and destination ports are a 16-bit filed each, which containing a transmitting system unique value assigned at the time the transport connection is created. These ports identify sending and receiving applications. The two values are used as the transport addresses, together with the source and destination IP addresses in the IP header.

- Sequence number- is the number of data packets in a series of segment. The sequence number is a 32-bit unsigned number.
- Payload length- indicates the length of the elements or data part in bytes following the fixed header.
- Flag- Flag bit, 'f' has a different purpose, depending on the packet types. Table (2.11) shows the flag indications.
- Reserve filed- if the reserve field exists, the value must always have a value of zero.

PACKET TYPES	FLAG	INDICATION
DT	F=1	end of stream
JC	F=1	join request acceptance
	F=0	join request rejection
TC	F=1	join request acceptance
	F=0	join request rejection
ND	F=1	connection pause period
	F=0	other cases
LR	F=1	user-invoked leave
	F=0	troublemaker ejection
CT	F=1	an abnormal termination
	F=0	normal termination
Other packets	field is ignored	

Table (2.11): ECTP Control Packet's Flag

2. EXTENSION ELEMENTS (DATA PARTS)

Extension elements have three parameters, which are described as follows:

- **CONNECTION INFORMATION-** 8-byte extension elements contain of information on the multicast transport connection. Figure (2.10) shows the connection information element fields.

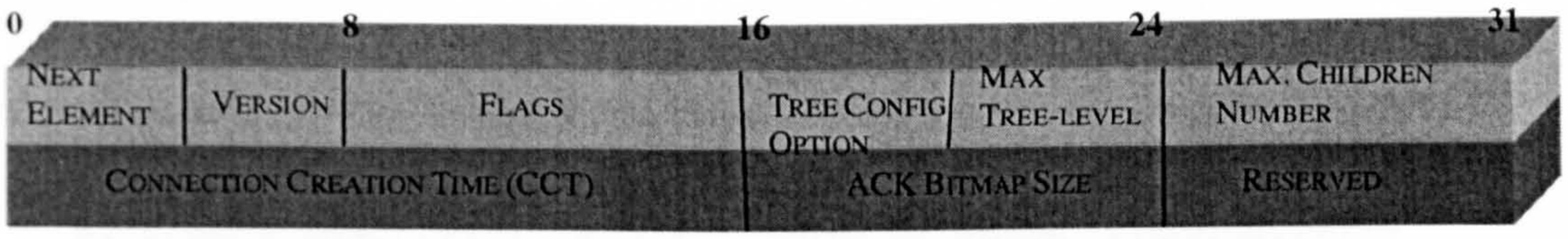


Figure (2.10)- Connection Information Element

Table (2.12) shows the specification of connection information element:

CONNECTION INFORMATION ELEMENT	DESCRIPTION
Next element	indicates the type of the next element immediately following this element
Version	defines the version of this element. The default value is '1'
Flags	consists of Connection type (CT) specifies which type of connection is being established.
Tree configuration option	specifies the tree configuration option used in the connection.
Maximum tree level (MTL)-	specifies the maximum number of tree levels for control tree.
Maximum Children Number (MCN)-	specifies the maximum number of children nodes which a parent can keep on the control tree.
Connection Creation Time (CCT)-	specifies a timer to limit the connection creation in units of 10 milliseconds. If this timer expires, TO completes the connection creation even if some of the children have not responded with CC packets. This timer is also used as a basis for LO to calculate its tree creation time.
ACK bitmap size	specifies the size of the bitmap in the acknowledgement element, in units of word. This value is not subject to negotiation, thus all receiver nodes must configure the bitmap field in the acknowledgment element, based on the advertised ACK bitmap size.

Table (2.12): Specification of connection information element

- **TREE MEMBERSHIP ELEMENT**- 20-byte extension elements contain of information on the local groups. Figure (2.11) shows the tree membership element fields.

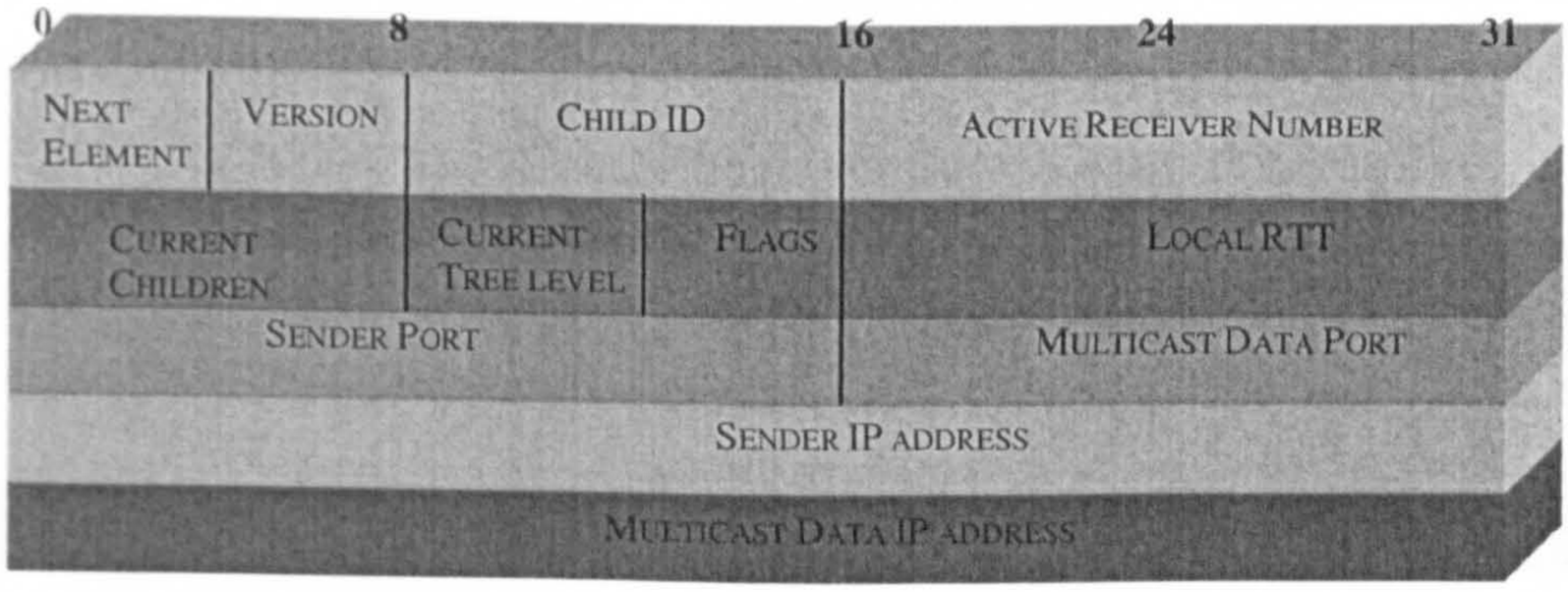


Figure (2.11)- Tree membership element

TREE MEMBERSHIP ELEMENT	DESCRIPTION
Next element	indicates the type of the next element immediately following this element.
Version-	defines the version of this element. The default value is '1'.
Child ID	specifies the ID number of child, which is assigned by its parent in the tree configuration.
Active Receiver Number (ARN)-	is a number of active descendants. Each LE sets the ARN to '1' and the parent LO aggregates ARN values for its children.
Current Children Number (CCN)-	is a number of active children for a LO. Each LE sets CCN to '0' value.
Current Tree Level (CTL)	specifies the current tree level. TO is in the level 0 and its children are in the level 1 (levels options).
Flags-	indicating the receiver is LO or LE.
Local RTT	indicates the round trip time for a local group in units of 10 milliseconds.
Sender port	indicates the port number of ECTP sender (TO).
Multicast data part	indicates the port number of the multicast channel
Sender IP address	indicates the IPv4 address of ECTP sender (TO)
Multicast data address	indicates the IPv4 address of the multicast data channel.

Table (2.13): Specification of tree membership element

- **ACKNOWLEDGEMENT ELEMENT**- this element provides information on error, flow and congestion controls, which consists of fixed 8-byte and the variable-sized bitmap that depends on the ACK bitmap size. Figure (2.12) shows the acknowledgement element fields.

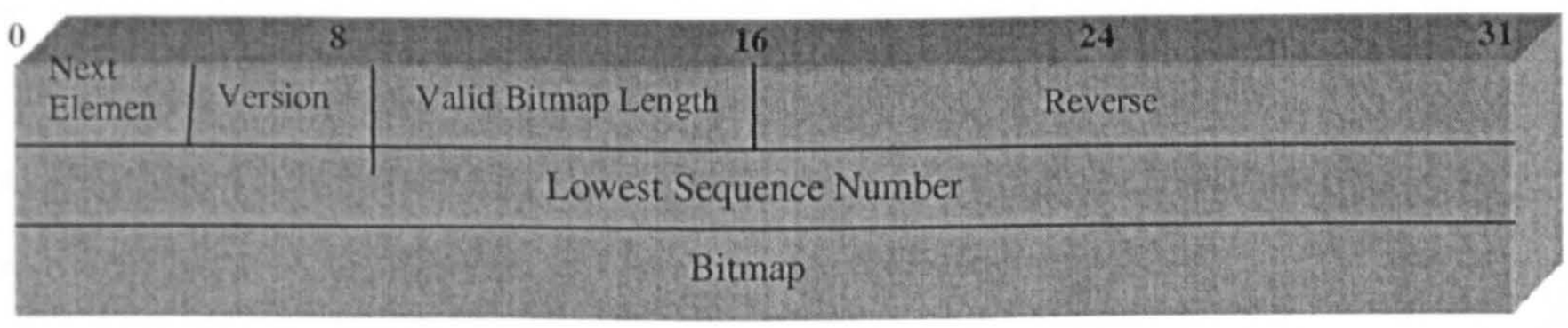


Figure (2.12)-Acknowledgement Element

ACKNOWLEDGMENT ELEMENT	DESCRIPTION
Next element	indicates the type of the next element immediately following this element.
Version-	defines the version of this element. The default value is '1'.
Valid bitmap length	indicates the length of valid bitmap
Lowest Sequence Number (LSN)	is a sequence number of the lowest numbered data packet which has not been yet received
Bitmap	indicates which data packets have been lost. It contains valid bitmap length bit; starting from the LSN sequenced number.

Table (2.14): Specification of acknowledgment element

- **TIMESTAMP ELEMENT**

  The Network Timestamp Protocol (NTP) is used for specifying timestamps [Mills89]. NTP timestamps are represents as a 64-bit unsigned fixed-point number. The integer part is in the first 32 bits and the fraction part is 32 bits. If the NTP system is not used, ECTP uses the TCP timestamp calculation algorithm. In this case, only the first 32 bits for integer part will be valid. A flag bit is employed to indicate which timestamp mechanism is used. Figure (2.13) shows the timestamp element structure.

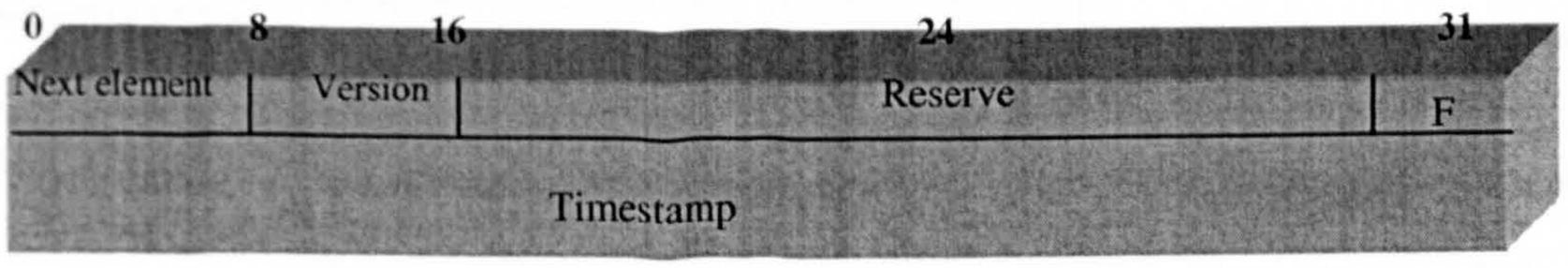


Figure (2.13)- Timestamp Element

The timestamp elements are specified as follow:

- Next element- indicates the type of the next element immediately following this element.
- Version-defines the version of this element. The default value is '1'.
- Flag (F)- is set to '0' for use of TCP timestamp mechanism,
- Timestamp- contain timestamp value

2.7.6 PACKET STRUCTURE

The following tables categorized the packet structures in three different formats. Table (2.15) shows the packet structure that uses only the fixed header. [CP+00], [Koh01]

PACKET STRUCTURE		PACKET FORMAT
NULL DATA (ND)	When To has no data to transmit or it is in the connection pause period, it transmits an ND packet every null data time (NDT) interval	ND= Fixed header
LATE JOIN REQUEST (JR)	A late joiner can join the connection by sending a JR packet to TO.	JR = Fixed header
TREE JOIN REQUEST (TJ)	A receiver joins a control tree by sending a TJ packet to an on-tree LO during tree configuration phase.	TJ = Fixed header
LEAVE REQUEST (LR)	When a child wants to leave the connection, it sends a LR packet to its parent. LR also used for a parent to eject a trouble-making child	LR= Fixed header
CONNECTION TERMINATION (CT)	TO terminations the connection by sending a CT packet to all receivers.	CT = Fixed header

Table (2.15): Fixed header Packet Structure

Table (2.16) shows the packet structure that uses the fixed header and the data part.

PACKET STRUCTURE		PACKET FORMAT
RETRANSMISSION DATA (RD)	Each parent LO transmits the RD packets to its children over the multicast control address	RD= Fixed header + Data part
RETRANSMISSION DATA (RD)	Each parent LO transmits the RD packets to its children over the multicast control address	RD= Fixed header + Data part
DATA (DT)	TO transmits a multicast stream with DT packets.	DT = Fixed header + Data part

Table (2.16): Fixed header and Data part packet structure

Table (2.17) shows the packet structure that uses the fixed header and the extension elements.

PACKET STRUCTURE		PACKET FORMAT
CREATION REQUEST (CR)	TO creates an ECTP connection by sending a CR packet to the all receivers over the multicast data address.	CR = Fix header + Connection information element
CREATION CONFIRM (CC)	In the response to the CR packet, each receiver sends a CC packet to its parent by unicast.	CC = Fixed header + Tree membership element
TREE JOIN CONFIRM (TC)	An on-tree TO or LO responds with a TC packet to the TJ packet.	TC = Fixed header + Tree membership element
ACKNOWLEDGEMENT (ACK)	Each child sends an ACK packet to its parent by unicast.	ACK = Fixed header + Tree membership element + Acknowledgement
HEARTBEAT (HB)	Each parent LO sends a HB packet to its children every Heartbeat generation time (HGI) interval.	HB= Fixed header + Tree membership element
LATE JOIN CONFIRM (JC)	TO responds with a JC packet to the JR packet.	JC = Fixed header + Connection information element

Table (2.17): Fixed header and the extension elements packet structure

2.7.7 PROTOCOL PROCEDURES

1. **ENROLMENT BEFORE CONNECTION CREATION**

   It is assumed that the enrolment procedures have been processed before the connection creation and all the active receivers have been enrolled into their multicast groups. This procedure has been discussed in details in Appendix B. [PC+01]

2. **CONNECTION CREATION**

   - **Connection creation procedures/ operations**

   The connection creation procedure can be summarized as follows [PE+01], [Koh01]:

   1. TO transmit a CR packet to all receivers by multicast, then TO activates the *Connection Creation Time* (CCT) timer.
   2. When a receiver receives the CR packet, it begins to configure the control tree.
   3. After each receiver joins the tree, it sends a CC packet to its parent by unicast, and then waits for multicast data from TO. Each on-tree parent LO aggregates the CC packets from all of the children and then sends them to its parent.
   4. TO aggregates CC packets from its children while the CCT timer is valid. If the CCT timer expires, TO completes the connection creation for the receivers that have sent CC packets until then.

   After the connection creation is completed, TO transmits multicast data. The receivers that have not participated in the connection may join the connection as late-joiners.

   - **Control tree creation**

     In the connection creation, ECTP configures a hierarchical control tree connecting TO and LEs via LOs. ECTP provides two options for control tree creation [PK+00]:

OPTION (1)- Level-1 configuration, where no LO is employed. Figure (2.14) illustrates no LOs is employed in the control tree. In fact no tree is configured. All the receivers become the children of TO. This option may be used in multicast connections that do not display scalability problems.

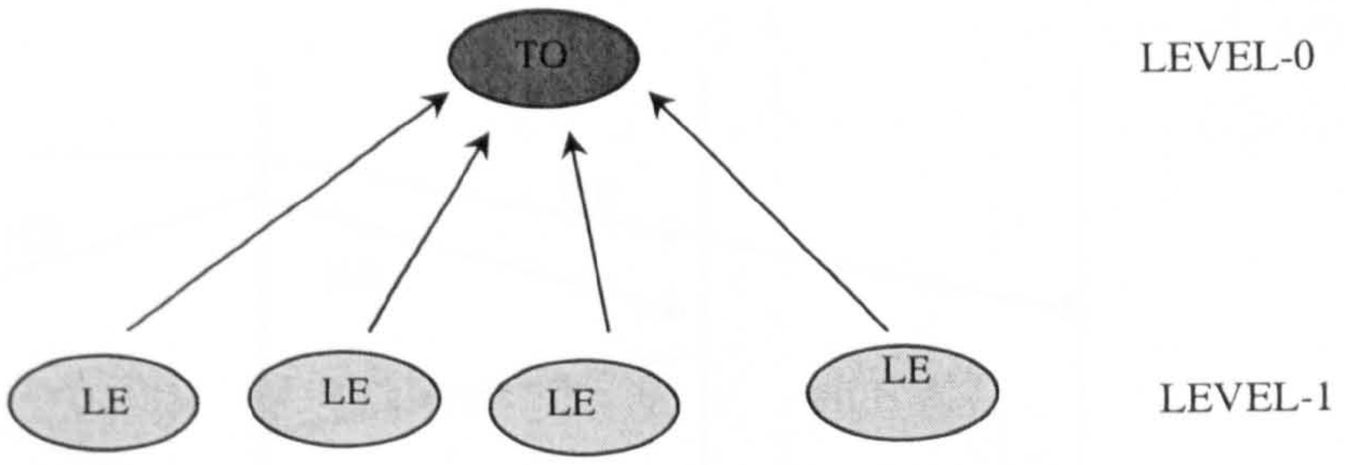


Figure (2.14)- Level-1 Tree control, option (1)

OPTION (2)- Level-2 configuration, where all LOs are connected to TO. Figure (2.15) illustrates all LOs are connected to TO, and each LE can join an on-tree LO or TO.

Figure (2.15)- Level-2 Tree control option (2)

Each receiver node (LO or LE) starts the tree creation as soon as a CR packet is received. One of these options must be specified in the *connection information element* for stable operation and maintenance in ECTP protocol mechanisms. Figure (2.16) shows the Tree creation procedures in the connection creation operation. [KP+00]

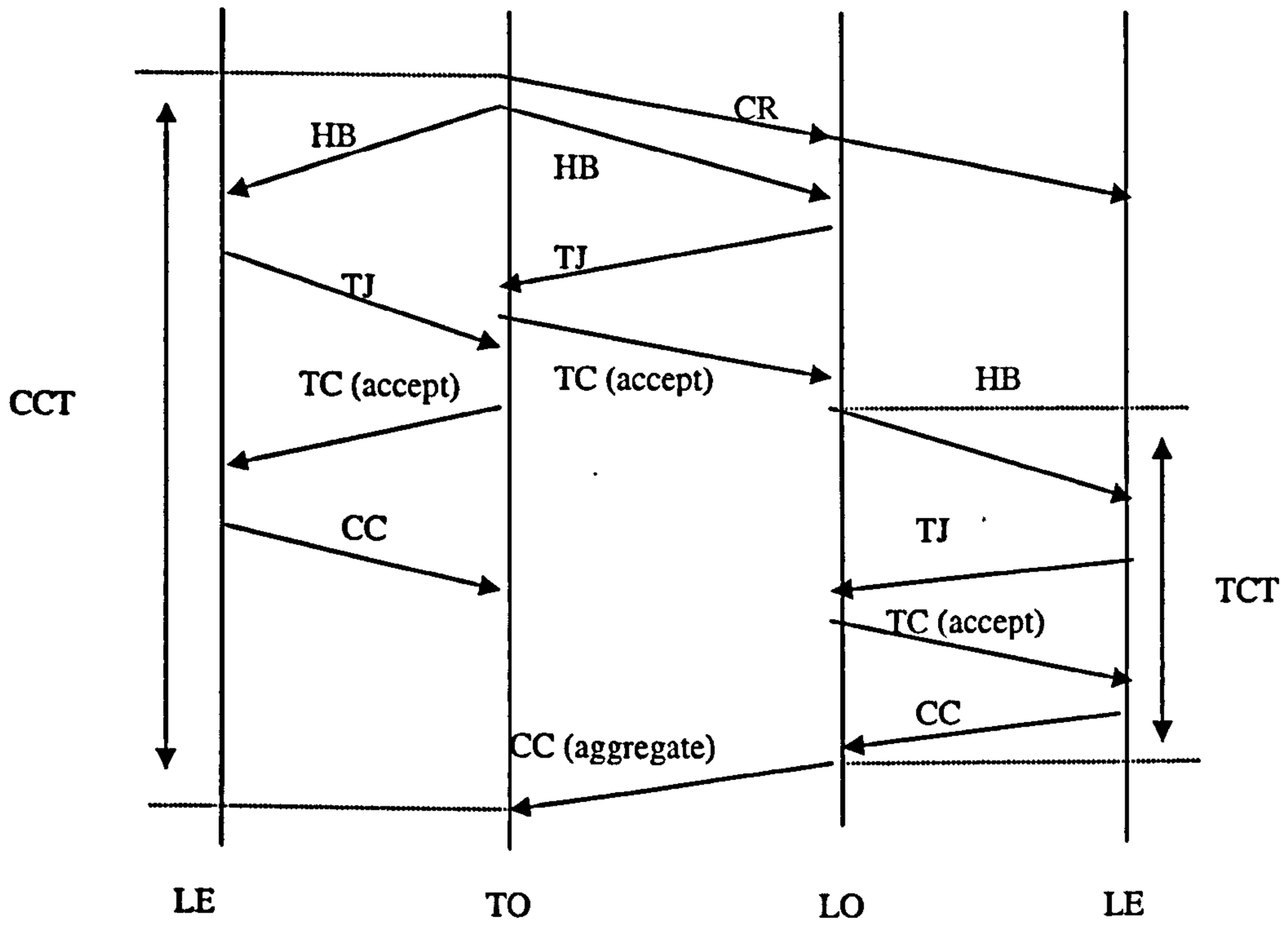


Figure (2.16)- Tree creation procedures

3. DATA TRANSMISSION

After the connection is created, TO transmits multicast data to all the receivers. TO will generate DT packets by the segmentation procedure. It happens when TO splits a multicast data stream into multiple DT packets where each DT packet has its own sequence numbers.

When TO has no data to transmit, TO transmits the periodic ND packets. ND time (NDT) is a time interval between multicast transmissions of ND packets by TO.

The NDT timer is activated after the connection is created. Each time a DT or RD packet is transmitted, the NDT timer is reset. If NDT timer expires, TO transmits a ND packet.

All the received data packets are delivered to the application in the order sent by TO. Each receiver reassembles the received packets. Corrupted and lost packets are detected by using a checksum and sequence number. All of the lost DT packets (include corrupted) are recovered in the error control function. [EC00], [Koh01]

- **CHECKSUM**

  Checksum function is used to detect the corrupted received packet. It covers a whole packet format including header and extension elements (data part). The checksum must be applied to all packet types. [Koh01]

- **SEQUENCE NUMBER**

  TO is sequentially numbered a new DT packet, where each sequence number is used to detect lost data packets by receivers and to manage the transmission and retransmission buffer by TO and LOs. [Koh01]

4. **ERROR RECOVERY**

   The reliability control mechanisms typically consist of error recovery, flow and congestion control. Flow and congestion control mechanisms are designed to adjust the data transmission rate base on the monitored status of the receivers and the network, which are defined in the QoS management functionality (chapter 4). This section focuses on error recovery, which consists of error detection by receivers, retransmission request by receivers via ACK packet and retransmissions by parents. In this work, we assume that the ECTP sender transmits only multicast data at fixed rate. [EC00], [Koh01], [CE98]

5. CONNECTION PAUSE AND RESUME

This function is used to suspend multicast data transmission temporarily. If the connection pause is indicated, TO transmit periodic ND packets with the Flag bit of fixed header set to '1'. In connection pause period, TO must not transmit any new DT packet, while control packets including RD and HB can be sent. [EC00],[Koh01]

6. LATE JOIN

The following procedures is the performance of the late-joiner receivers when they want to join an existing connection [EC00],[Koh01]:

1. The late-joiner sends a JR packet to TO by unicast. It then activates Retransmission Time (RXT) timer,
2. One reception of a JR packet, TO responds with a JC packet to the late-joiner.
3. If the late-joiner receives a JC packet with an acceptance flag within RXT, then it begins to join the control tree in the tree configuration.

7. LEAVE

This function is used when a receiver leaves an existing connection, or parent ejects a trouble-making child. [EC00], [Koh01]

8. TREE MEMBERSHIP MAINTENANCE

After an initial control tree is created in the connection creation, the tree membership is maintained until the connection is terminated. [PC+01]

- **Tree configuration for late joiners**

  After the connection is created, each LO advertises periodic HB packets over its multicast control address. When the late-joiner receives a JC packet from TO, it begins to locate a suitable parent.

The late-joiner listens to HB packets from one or more LOs, and information on candidate parents are recorded into the parent list.

- **Tree reconfiguration for leaving receivers**

  When a child leaves the connection, the parent removes the child out of its child list.

- **Tree reconfiguration against node failures**

  To detect a node failure, the node failure threshold (NFT) is employed. The tree maintenance procedures are different for the node types: TO, LO and LE.

9. CONNECTION TERMINATION

TO terminates a multicast transport connection by sending a CT packet to all receivers by multicast. When the connection termination is indicated, TO shall discard all subsequently received packets and also freezes the local port number. One the receipt of a CT packet, each receiver freezes the local port number. This function will be initiated after all the multicast data are transmitted. TO also terminates the connection on detection of fatal protocol error. [Koh01]

2.8 SUMMARY

Multicast service and applications: types and classifications have been discussed in this chapter. It has also explained about the multicast applications requirements. This chapter has investigated the various approaches for multicast protocols and as result, it has been shown that the tree-based protocol is more reliable and scalable compare to other exciting multicast protocol approaches (sender-initiated, receiver-initiated, and ring-based).

Finally, Enhance communication transport protocol (ECTP), control part specification has been discussed. ECTP features "tight management of multicast sessions" from the sender's perspective. Tight management in the multicast sessions happens when the sender is at the heart of group communications. The sender can perform overall management of the multicast connection establishment; group membership and Quality of Services perceived by end users with the help of Internet Group Management Protocol (IGMP) and IP multicast routing protocols. The design of ECTP uses the combination of sender-initiation and tree-base approaches to support multicast applications in the simplex connection.

Chapter 3

Quality of Service

3. QUALITY OF SERVICE (QOS)

3.1 OVERVIEW

Today's Internet provides only *Best Effort* service, i.e. the traffic is processed as quickly as possible but there is no guarantee for *Quality of Service* (QoS). QoS was first introduced to describe characteristics of low level data transmission in communication systems. With the appearance of multicast service and their applications [Miller99], the meaning of QoS was broadened to cover all system components in a distributed multicast system. QoS can represent the set of those quantitative and qualitative characteristics of a distributed system necessary to achieve the required functionality and performance of an application. Functionality and performance include both the presentation of data to the user and general user satisfaction. [MCDysan00]

The QoS is expressed using *QoS parameters*, which are described as specific attributes of multicast service, each of which have target values. The structure of a QoS parameter consists of an *attribute name* and *target value*. An attribute describes a single aspect of the functionality or performance of a distributed multicast service. Due to the very nature of multicast applications, a vast set of different QoS parameters exits, such as throughput, delay, etc, where each of these has a specific target value(s). Applications have to specify the QoS level they need, such that the multicast service is able to provide the respective functionality and performance. The requirements of an application are expressed as a set of QoS parameters that are used to describe the service level that it requests of the underlying communication framework. [VK99]

ATM was first networking technology designed with QoS capabilities; however the concept of QoS in ATM is rather complex. Therefore, the ATM Forum [Dobrowski98] has defined the simpler notion of QoS categories tailored to broad classes of applications in Internet protocol (IP) [JN98].

The concept of QoS per connection has no meaning in a connectionless protocol i.e. IP. Therefore, Internet engineering task force (IETF), [IETF], has introduced several QoS

services architectures to adopt the idea of QoS for IP networks. By now there are two QoS architectures which have been defined: *Integrated services* (IntServ) (section 3.4) and *Differentiated services* (section 3.6) (DiffServ) [Stalling98].

3.2 QoS MANAGEMENT PARAMETERS AND COMPONENTS

QoS management is concerned with the support of QoS parameters to be maintained within certain limits between two or more users. [TW97]

3.2.1 PARAMETERS

As it discussed in section (2.4.3), Multicast applications have varying performance requirements. Therefore it is necessary to design a service that provides guaranteed QoS satisfaction. These requirements have been defined, as QoS parameters that the applications are interested in and the range of QoS desired for each chosen parameter [QA01]. Table (3.1) shows the common QoS parameters (section 2.4.3.1) that most multicast applications are generally concerned with.

QoS PARAMETERS	
Throughput	Represents an amount of application data over a specific time period
Transit delay	Represents end-to-end transmission time from a sender to a receiver
Transit jitter	Represents the variation of network delay over short time scales
Data loss	Represents the ratio of the amount of lost data to the amount of transmitted data.

Table (3.1): QoS parameters

3.2.2 COMPONENTS

QoS management is the concrete realization of required levels of QoS in a real environment. The components of QoS management have been summarized as follows [TW97]:

- **QOS NEGOTIATION**

  QoS negotiation is defined as a *static* QoS management component. The QoS negotiation function is responsible for analyzing an activity and finding a composite of the individual QoS levels supportable by applications, which is necessary and sufficient to realize the QoS of the complete activity.

- **QOS MAINTENANCE**

  QoS maintenance is defined as a *dynamic* QoS management component. To maintain an agreed level of QoS it is often not sufficient to dedicate resource to an activity at QoS negotiation time. Instead, dynamic QoS maintenance is frequently required to ensure that the required performance of individual system components is kept within bounds. The QoS maintenance function is particularly important in cases where resources must be multiplexed among activities.

- **QOS MONITORING**

  QoS monitoring is defined as a *dynamic* QoS management component. QoS monitoring is used to allow each level of system to track the ongoing QoS levels achieved by the lower layers and compare them with the initial requirement. It often takes the form of a feedback loop, monitoring the QoS being achieved by the monitored component, comparing it against the target and then performs if necessary.

3.3 QoS SERVICES

The network needs only to provide a certain level of quality for the packet that conforms to the QoS parameters. IETF has defined two QoS services architectures: *Integrated services* and *Differentiated services*.

3.4 INTEGRATED SERVICES (INTSERV)

When the first Integrated Services (IntServ) session was held in the November 1993 the main concern was to discuss how the audio and video casts from IETF meetings could be supported by the network [Kilkki99, p.53]. From the very beginning the main principle behind Integrated Services was to enable the transmission of real-time data without the need to modify the underlying Internet architecture. The goals of the IntServ working group were [IETF-IntServ]:

- to clearly define the services which this enhanced model provides,
- to define the interfaces between the application and network service, routers and sub-networks,
- to develop router validation requirements in order to ensure that the proper service is provided.

3.4.1 INTSERV ARCHITECTURE

The fundamental idea behind integrated services is that the existing Internet architecture does not have to be modified. IntServ architecture is defined as a set of extensions to the traditional best effort model of the Internet with the goal of allowing end-to-end QoS to be provided to the multicast applications. The two main building blocks of IntServ are *resource reservation* and *admission control*. IntServ assumes that the guarantees for the delivery of multicast applications can be achieved only by reserving resources from the network nodes. This requires flow-specific states in the network nodes and represents a fundamental change in the traditional network node model [EA+00].

It is quite obvious that some kind of explicit setup mechanism is needed to create and maintain flow-specific reservations in the network. The main components of IntServ are a reservation setup protocol known as Resource Reservation protocol (RSVP) and admission control that determines whether a new flow can be granted for the requested resource reservation without affecting the exciting established flows. [Iyer97]

3.4.2 CLASSIFICATION

IntServ has been defined in three classes, which are described as follow:

- **BEST-EFFORT**

Best-effort is the default and traditional QoS service. This service does not guarantee any kind of QoS; it only delivers the packets as fast as possible. The most commonly used scheduling mechanism is first in first out (FIFO) (section 3.7), that delivers packet to the receiver in the same order as they came from the sender. This service is sufficient for all application if the network element is lightly loaded. In a congested network delay and dropping packets are typical and therefore best-effort services are not acceptable for multicast application. The best-effort service does not use any traffic descriptors such as Traffic (TSpec) and Reservation (RSpec) specifications (section 5.5.6) [Shenker95].

- **CONTROLLED- LOAD NETWORK ELEMENT SERVICE**

The controlled-load service is designed to provide end-to-end traffic behaviour that closely approximates traditional best-effort services in unloaded network conditions. In other words, the service is better than that which best-effort can offer in a congested network. The applications using this service may assume that a high percentage of the transmitted packets will be successfully delivered by the network, i.e. the ratio of dropped packets should be equivalent to the error rate of the transmission media. The latency introduced into the network will not greatly

exceed the minimum delay experienced by any successfully transmitted packet. The controlled-load service does not make use of specific target values for such control parameters as delay or loss [Wroclawski97].

The controlled-load service uses the admission control to assure that service is received even when the network element is overloaded. The receiver's application requests the controlled-load service with the help of the sender's TSpec. The reservation request will be accepted if the admission control process is successful, i.e. if the network element has the capacity to forward the flow's packets as the FLOWSPEC assumes and if the receiver has permission to make such a reservation. The controlled-load service is provided for traffic that conforms to the traffic specification given at flow set-up time. The amount of data sent does not exceed $r*T+b$, where r and b are the token bucket parameters and T is the length of the time period. The non-conforming packets can be forwarded as best effort traffic [JS+92].

The aim of the controlled-load service is to provide a better service than best effort but without any hard guarantees. As has already been mentioned, this service is sufficient for applications, which can operate with best-effort service in a lightly loaded network, but not in a congested one [Wroclawski97].

- **GUARANTEED SERVICES**

The Guaranteed service provides firm end-to-end QoS, which guarantees both delay and throughput. This service describes both traffic descriptions: Traffic specification (TSpec) used to introduce the flow's traffic characteristics to the network, and Reservation specification (RSpec) used by the flow to request a specific requirement from the network, e.g. bounded delay.

Using these traffic descriptions, the network can compute various parameters describing how it will handle the flow, and by combining the parameters, it is possible to compute the maximum delay a packet can experience. This service does not control the minimum or average delay (jitter) of datagrams, merely the maximum queuing delay.

The service guarantees that datagrams will arrive within the requested delivery time and will not be discarded due to queue overflows provide the flow's traffic stays within its specified traffic parameters. This service is intend for applications which need a firm guarantee that a datagram will arrive no later than a certain delay bound.

Note that this service does not attempt to minimize the delay jitter, but merely controls the maximal queuing delay. The guaranteed service policing are out of scope this work.

RFC 2212 has specified that: "*The data transmission in the IP network causes two types of delay: A fixed delay and a variable queuing delay. The fixed delay includes transmission delays, the properties of chosen path. Therefore only the queuing can be controlled by guaranteed service. The guaranteed service framework asserts mathematically that the queuing delay is a function of two factors: The token bucket depth (b) and token data rate (r). This model is the service that would be provided if there was a wire of bandwidth R between source and receiver*". Thus the end-to-end delay bound in guaranteed service that has been defined in RFC 2212 are as follows:

$$\left[\left(\frac{b-M}{R}\right)\times\frac{(p-R)}{(p-r)}\right]+\frac{(M+Ctot)}{R}+Dtot \qquad (r \le R < p) \qquad (3.1)$$

or in a case where the peak rate is smaller that data rate R is:

$$\left[\frac{(M+Ctot)}{R}\right]+Dtot \qquad (r \le p \le R) \qquad (3.2)$$

where r, b, p and M are token bucket parameters (section 3.4.3.1), R is data rate and must be greater than or equal to the token-bucket data rate (r). [CS+92]

The rate-dependent error term *Ctot* represents the end-to-end sum of delays that the datagram might experience due to the rate parameters of the flow, also known as packet serialisation. The rate-independent error term *Dtot* represents the worst case of end-to-end sum of delays in the transit time as the datagram in transferred

through the service element, i.e. a router. Table (3.2) summarises the IntServ classes and their performances. [SP+97]

QOS SERVICES		PERFORMANCE	QOS RELIABILITY
Integrated services	Best-Effort	QoS is ignored in providing service No traffic descriptors	Low
	Guaranteed	QoS is guaranteed, Using TSpec, RSpec	High
	Controlled - load	QoS is guaranteed, no guaranteed delay, Using TSpec	High

Table (3.2): IntServ classes and their performances

3.4.3 INTSERV TRAFFIC COMPONENTS

IntServ consists of two traffic components: *Traffic control* and *Reservation setup*. Traffic control is implemented inside each of the network nodes (routers or switches) that manages and regulates the data packets; Reservation setup is an end-to-end message that carries the application's desired QoS requirement and provides the network feedback to the application. Figure (3.1) shows the structure of IntServ components [BC+94].

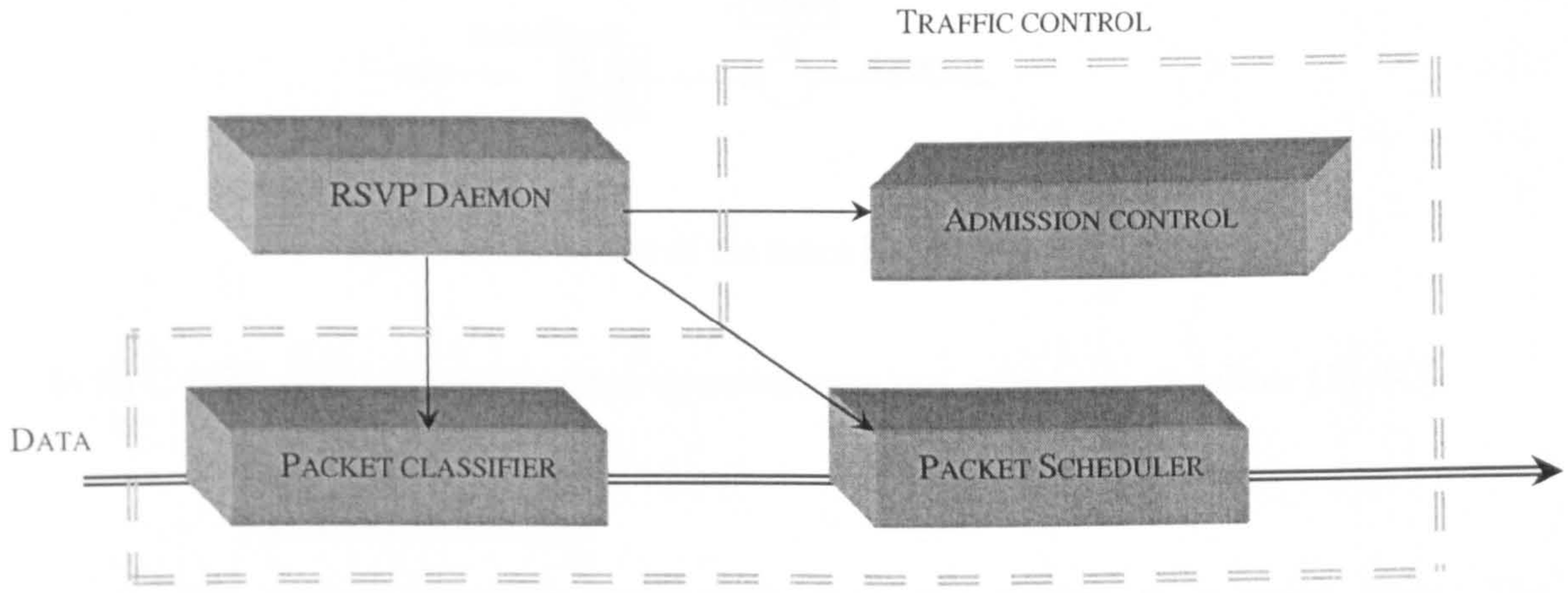


Figure (3.1): Structure of IntServ components

3.4.3.1 TRAFFIC CONTROL

The traffic control mechanism uses a token bucket approach that dictates when traffic can be transmitted. The transmission is authorised by on the presence of tokens in the bucket.

The token bucket is a very useful mechanism in policing data flow. This mechanism can also be used in the Differentiated Services, where the policed traffic flow is aggregated, not individual [BC+94].

The token bucket contains tokens, each of which represents a unit of bytes. When the tokens are present, a flow is allowed to transmit traffic. If the bucket contains no tokens, a flow cannot transmit packets. Tokens are continuously generated at rate r, which in the case of IntServ is measured in bytes of IP datagrams per second. When the IP packet of size L is transmitted, the corresponding amounts of tokens are decrement from the bucket. The bucket depth b, measured in bytes, determines how bursty the transmitted flow can be. If there are tokens in the bucket it is possible to transmit traffic faster than the token rate assumes, but in the long run the average rate can not be higher than the token rate r. Figure (3.2) shows the nature of the token bucket scheme. [FH98].

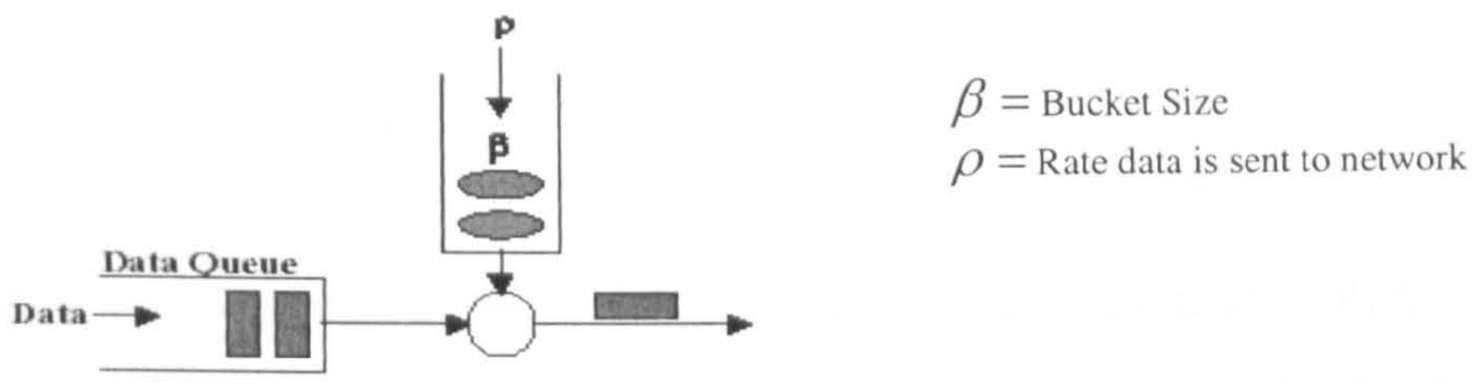


Figure (3.2): Nature of the token bucket scheme

In RFC 1633, the traffic control has three components, which are as follow [JS+92]:

- **PACKET CLASSIFIER**

  The packet classifier maps each incoming packet to a same class. The classification may be based on the contents of the packet header, such as source and destination addresses. All packets in the same class get the same treatment from the packet scheduler.

- **PACKET SCHEDULER**

The packet scheduler manages the forwarding of different packet streams using queuing mechanism: Priority scheme or Weighted Fair queuing (section 3.7). To achieve the reserved QoS service, the packet scheduler must be implemented at the point where packets are queued.

- **ADMISSION CONTROL**

The admission control determines whether a new flow can be granted the requested QoS without impacting upon earlier guarantees. Admission control is invoked at each node to make a local accept/reject decision, at the time a host requests a real-time service along some path through the network. Admission control also controls accounting and administrative reporting. It can check if a node is allowed the reservation it is requesting and charge the user according to the QoS service.

3.4.3.2 RESERVATION SETUP PROTOCOL

The other component of the Integrated Services, in addition to the admission control, the packet scheduler and the classifier, is the reservation set-up protocol. Applications can negotiate their QoS requirements through reservation set-up protocol.
The reservation set-up protocol signals the negotiated QoS requirements over the network to create an end-to-end path. The reservation set-up protocol, which is designed for Integrated Services known as Resource Reservation Protocol, RSVP, [BZ+97].

3.5 RSVP PROTOCOL OVERVIEW

In general, the RSVP protocol is a signaling mechanism, which typically carries QoS requests from one end of network to the other, along the same path traversed by the data that requires QoS resources.

RSVP requests resources for simplex flows, i.e. it requests resources in only one direction. RSVP operates on the top of IPv4 [JN98] or IPv6 [Durand01], occupying the place of the transport protocol in the protocol stack. However, RSVP is not a transport, protocol but rather an Internet control protocol like IGMP [Fenner97] or routing protocols [Moy97].

RSVP is receiver-oriented, i.e. it is the receiver who is responsible for requesting specific QoS services and not the sender. This approach accommodates large groups, dynamic group membership and heterogeneous receiver requirements. Each RSVP sender host regularly transmits RSVP "Path" messages downstream along the uni-/multicast routes provided by the routing tables in the routers. Path messages store path information to each RSVP aware node in the traffic path. The path state includes at least the unicast IP address of the previous hop-node, which is used to route the "RESV" messages hop-by-hop in the reverse direction, as illustrated in Figure (3.3). Since the receiver is responsible for the reservation request, the RESV message specifies the desired QoS and sets up the reservation state to each node in the traffic path. The reservation is made in one direction only, but an application can act both as a sender and receiver at the same time [White97].

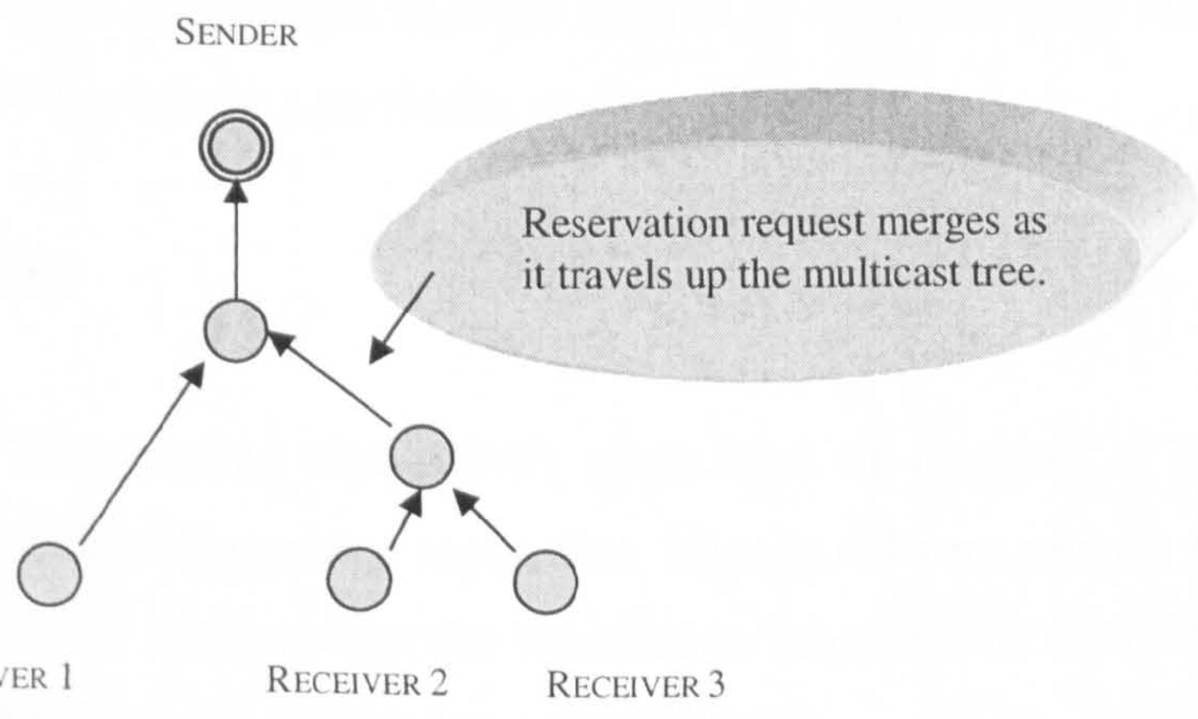


Figure (3.3): RESV messages hop-by-hop

At each RSVP aware node an RSVP request is passed to two local decision modules: admission control and policy control. The policy control checks if the user has an administrative permission to make a reservation. The admission control determines if the node has enough free resources to supply the requested QoS. If both checks succeed, the parameters from the Filter specification and Flow specification are set in the packet classifier and packet scheduler, respectively. This is shown in Figure (3.4). The admission control, packet classifier and packet scheduler together are called the traffic control. [ZD+93]

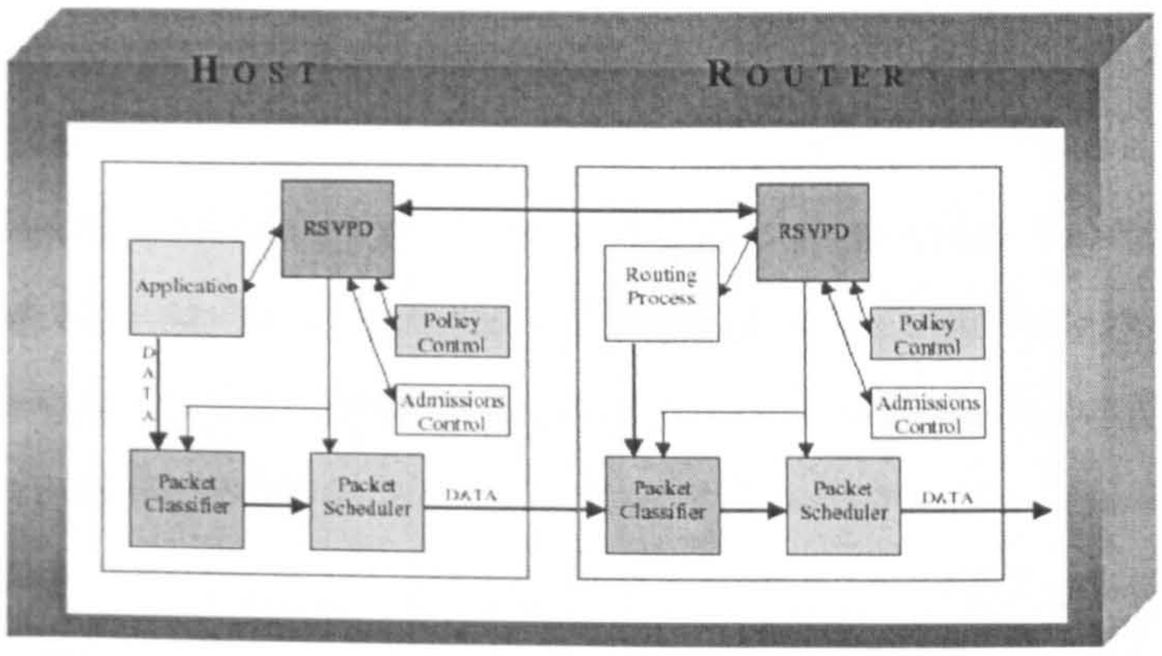


Figure (3.4): RSVP protocol components (source from RFC 2205)

3.5.1 FLOW AND FILTER SPECIFICATIONS FLOWS

In general, an RSVP reservation request specifies the amount of resources to be reserved for all or some subnets of the packets in a particular session. [Pessi97]

3.5.2 RESERVATION STYLES

Due to the interaction of multicasting and RSVP, some kind of merging of RSVP reservation is required. Multicast uses packet replication when it delivers packets to the different next-hop-nodes. RSVP must merge the reservation request and compute the maximum of their FLOWSPEC at each replication point. [BZ+97]

The reservation style is one type of information included in the RESV messages. The reservation styles concern the treatment of reservation for the different senders within the same session (multipoint-to-multipoint) and the selection of senders. RFC 2205 defined three reservation styles, which are as follow [ZD+93]:

- **WILDCARD-FILTER (WF)** style creates a single reservation that is shared by the flows from all upstream senders. This reservation is the largest of the resource requests from all receivers. Senders are not explicitly selected and therefore a new sender can automatically join the session.

- **FIXED-FILTER (FF)** style creates a distinct reservation for data packets from a particular sender. Consequently, the resources are not shared with other senders' packets for the same session. Also the senders must be explicitly selected, which is done by the filter spec.

- **SHARED-EXPLICIT (SE)** style creates one shared reservation, just like WF, but the receiver is allowed to explicitly specify the set of senders to be included.

Table (3.3) summarises the reservation styles.

SENDER SELECTION	DISTINCT	SHARED
Explicit	Fixed-filter (FF) style	Shared-explicit (SE) style
Wildcard	None defined	Wildcard-Filter (WF) Style

Table (3.3): Reservation styles (source RFC 2205)

3.5.3 RECEIVER INITIATION

RSVP uses receiver-initiation of reservation, which a receiver is assumed to learn the senders' offered FLOWSPEC by the higher-level mechanism. It then generates its own desired flow spec and propagates it towards the senders, making reservation in each router along the way. [BZ+97]

3.5.4 SOFT STATE

There are two different possible styles for reservation setup protocols, in both of which multicast distribution is performed using flow-specific state in each router along the path [BZ+97]:

- **HARD STATE (HS) APPROACH**

  This approach is created and deleted in fully deterministic manner by cooperation among the routers. Once a host requests a session the network takes responsibility for creating and later destroying the necessary state. ST-II is an example of HS approach. [Topolcic90]

- **SOFT STATE (SS) APPROACH**

  RSVP takes this approach, which regards the Reservation State as cached information that is installed and periodically refreshed by the end hosts. If the rout changes, the refreshed messages automatically install the necessary state along the new route. The SS approach was chosen to obtain the simplicity and robustness that have been demonstrated by the connectionless protocols such as IP. [Clark88]

3.5.5 RSVP MESSAGES

RSVP has many messages types; the two most important are *Path* and *Resv* messages. The remaining RSVP message types are reservation teardowns (PATH-Tear and Resv-Tear), reservation errors (PATH-Err and Resv-Err) and the confirmation for a requested reservation (ResvConf). [BZ+97]

1. **PATH MESSAGE**

   An RSVP sender transmits PATH messages downstream to the unicast or multicast address. The PATH message contains information of the previous hop address and

three information elements called Sender Template, Sender TSpec, and ADSPEC. The Sender Template contains information called filter specification (filterspec), which uniquely identifies the sender's flow from other RSVP sessions on the same link. The identification is done by sender IP address and optionally the UDP/TCP sender port. The receiver uses this information to choose an appropriate size for the reservation request. The PATH message may also carry additional advertising specification (ADSPEC). ADSPEC is updated locally at every RSVP node. The information contained in the ADSPEC is divided into fragments: each fragment is associated with a specific control service. This way new service classes can be added in the future without modifying the RSVP protocol. The ADSPEC includes specific information about available resources, delay and bandwidth estimates, the indication of non-IntServ aware nodes and also service specific information for the guaranteed services. [BZ+97]

2. RESV MESSAGE

The RESV messages follow exactly the reverse path(s) the data packets will use, by using the path states created by PATH messages. RESV messages create and maintain reservation states in each node along the path(s). The RESV message contains information about the reservation style , the appropriate FLOWSPEC object, and the filterspec that identifies the sender. The format of the FLOWSPEC depends on the requested service. [BZ+97]

When a receiver requests controlled-load service, only a TSpec is contained in the FLOWSPEC. When requesting guaranteed service, both a TSpec and RSpec are contained in the FLOWSPEC object. The FLOWSPEC is used to set parameters in a node's packet scheduling process. The filter spec is in the same form as the Sender Template. The session is identified with the help of the destination address, protocol ID and destination port. The filter spec is used to set parameters in the packet-classifier process. Data that does not match any of the filter specs is treated as best-effort traffic. Figure (3.5) shows the Standard RSVP Path and RESV messages process. [Stuart99]

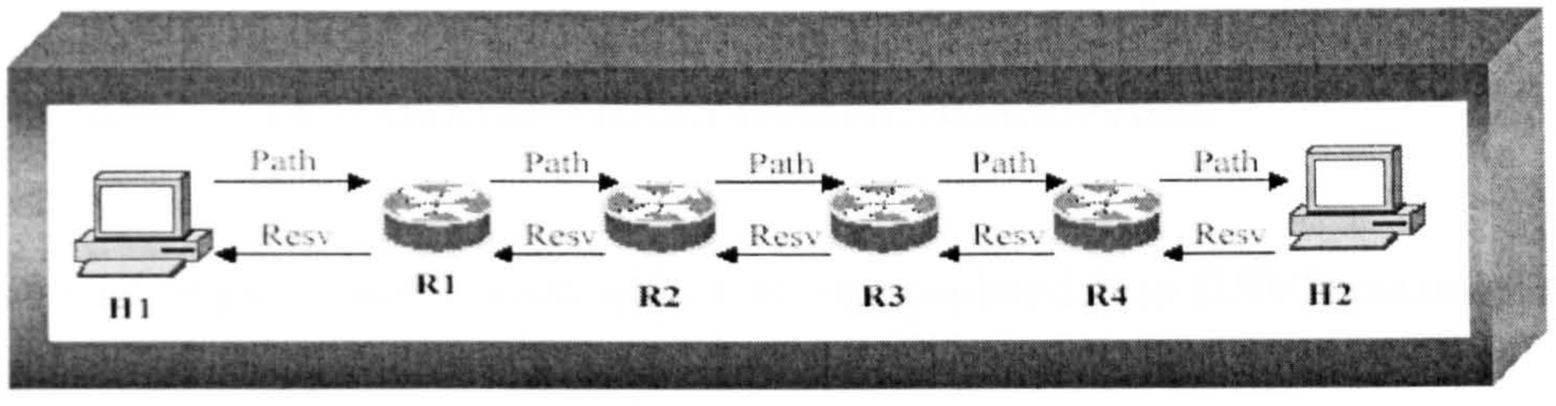


Figure (3.5): RSVP Path and RESV messages

3. TEARDOWN MESSAGE

RSVP "tear-down" (PATH-Tear and Resv-Tear messages) removes PATH or reservation state immediately. It is recommended that a reservation is torn down as soon as an application finishes, although is not necessary to do that explicitly. Teardown requests may be initiated also by a router as the reservation state times out. PATHTear messages are generated by senders or by routers due to the time-out of PATH State in any node along the traffic path. Similarly a ResvTear message is generated by receivers or by any node in the traffic path in which the Reservation State has timed out. [BZ+97]

4. TEARDOWN MESSAGE

PathErr and ResvErr messages are simply sent upstream flow to the sender that created the error. These messages do not modify the Path State in the nodes through which they pass. The PathErr and ResvErr indicate an error in the processing of PATH or RESV messages, respectively.

5. CONFIRMATION MESSAGE

When a receiver wants to obtain a conformation for its reservation request, it can include a confirmation request object in a RESV message. Each node in the transit path that receives a RESV message containing a reservation confirmation object sends a ResvConf message. Since the receipt of a ResvConf does not give any guarantees, a receiver may receive a ResvConf followed by a ResvErr. [BZ+97]

3.5.6 RSVP OBJECTS AND CORRESPONDING TRAFFIC DESCRIPTORS

RSVP has three objects, which each object is encapsulated into RSVP message types. They are described as follows [Stuart99]:

- **SENDER_TSPEC OBJECT**

The RSVP Sender_TSpec object carries information about a data source's generated traffic, which is usable by either Guaranteed or Controlled-load QoS control services. A PATH message must carry this object. The required RSVP Sender_ TSpec objects contain token bucket traffic parameters. TSpec uses a token-bucket algorithm to describe traffic parameters corresponding to a specific flow of IP packets. The token-bucket model is designed to smooth the flow of outgoing traffic and allows for data bursts higher send rates that last for short periods.

Data flow for RSVP sessions are characterized by the sender in the TSpec contained in PATH messages in RESV messages. Table (3.4) summarizes the token-bucket traffic descriptor in TSpec object.

TRAFFIC DESCRIPTOR	DESCRIPTION
Token Rate *(r)*	It is the continually sustainable bandwidth (byte/second) requirement for a flow. This reflects the average data rate into the bucket and the target shaped data tare out of the bucket.
Token-bucket depth *(b)*	The extent to which the data rate can exceed the sustainable average for short periods of time. More precisely, the amount of data sent couldn't exceed rT+b, where T is any time period.
Peak Rate *(p)*	This is set to the maximum send rate (bytes/second) if known and controlled. For all time periods (T), the amount of data sent cannot exceed M+pT.
Minimum policed size *(m)*	This should generally be set equal to the size of smallest packet generated by the application. This packet size includes the application data and all protocol headers above the IP level.
Maximum packet size *(M)*	This should be set to the size of the largest packet the application might wish to generate. This value must, by definition, be equal to or larger than the value of (m).

Table (3.4): Token-bucket traffic descriptor in TSpec object

- **FLOWSPEC**

The RSVP FLOWSPEC object carries information necessary to make reservation requests from the receiver(s) into the network. The object contained in RESV message. This includes an indication of which QoS control service is being requested, and the parameters needed for that service.

This object consists of Recevier_TSpec and Recevier_RSpec. The information in the FLOWSPEC flows upstream from data receivers toward data sources. It may be used or updated at intermediate network elements before arriving at the sending application.

Recevier_TSpec traffic descriptors are identical to Sender_TSpec. On the other hand, Recevier_RSpec includes the following traffic descriptors:

a) *R*: a reserved bandwidth,

b) *S*: a slack term.

where, *R* must be greater than or equal to *r* of TSpec,

the slack term *S* is in microseconds,

When a receiver generates an RSpec for a RESV message to be sent for a Guaranteed Service reservation request it must include a Slack Term, S (ms) as well as the amount of bandwidth, R to be included in each router along the path. S represents the amount by which the end-to-end delay bound will be below the end-to-end delay required by the application assuming each router along the path reserves R bandwidth according to the Guaranteed Service fluid approximation [BZ+97]. Regarding the calculation of the bandwidth reservation *R*, in [SP+97], "The end-to-end delay is bounded "by :

$$\left[\left(\frac{b-M}{R}\right)\times\frac{(p-R)}{(p-r)}\right]+\left[\frac{(M+C)}{(R+D)}\right] \qquad (r \le R < p) \qquad (3.3)$$

and

$$\frac{(M+C)}{(R+D)} \qquad p \le \mathrm{R}. \qquad (3.4)$$

- **ADSPEC object**

The ADSPEC (advertising specification) is an optional object descriptor that the sender may include in its generated Path messages in order to advertise to receivers the characteristics of the end-to-end communications path. Receivers determine the level of reservation required in order to achieving their desired end-to-end QoS can use this information. ADSPEC consists of a message header, a default general parameters part and Guaranteed Service part.

The RSVP ADSPEC is flowing downstream towards receivers, and may be used and updated inside the network before being delivered to receiving applications. This information includes both parameters describing the properties of the data path, including the availability of specific QoS control services, and parameters required by specific QoS control services to operate correctly. This object includes the following traffic descriptors: [BZ+97]

- Bandwidth of the link with the minimum available bandwidth on the end-to-end path.
- End-to-end transmission delays except queuing delay; these delays can be measured by two error terms, *C* and *D*.
    - Error term C is the rate-dependent error term. It represents the delay a packet in the flow might experience due to the rate parameters of the flow.
    - Error term *D* is the rate-independent, per-element error term. It represents the worst-case non-rate-based transit time variation through the service element. It is generally determined or set to the boot or configuration time.

3.5.7 AN ANALYSIS OF INTSERV AND RSVP

Integrated Services assumes that QoS can only be achieved by reserving resources for a particular flow. However, in the case of a backbone router, thousands of real-time sessions may exist simultaneously. The information about thousands of reservations needs to be stored, processed and accessed. Thus managing reservation states at the

routers with large interfaces causes significant overheads and degrades the performance of a router. In other words, the IntServ architecture has a scaling problem [CN97]. The RSVP signalling messages can also generate a considerable amount of control traffic when there are several reservations. Furthermore, the reservation set-up process may be delayed or the reservation may be timed out if the PATH or RESV messages are lost because of the congestion. This reduces the number of RSVP refresh messages. The latency and reliability problems are solved by acknowledgement messages. These messages can also be bundled, so that they will not produce more signalling traffic. There are two major drawbacks with Intserv architecture:

1. The hosts and applications should be RSVP aware, i.e. they have to generate RSVP messages and make resource requests. The RSVP aware performs dynamic and efficient control over the network. However at the moment there are only a few host and applications that are able to use RSVP. [XL99]

2. The scalability problems of IntServ could be solved with some kind of flow aggregation where several flows are treated as one. In the case of a backbone router only a small fixed number of service classes are offered [Detti99].

3.6 DIFFERENTIATED SERVICE (DIFFSERV)

Differentiated Services (DiffServ) is a direct extension to the work done by the Integrated Services working group. The goal is basically the same for IntServ architectures, to provide quality of service and service differentiation in IP networks. During the year 1997 the concern for the scalability of RSVP increased and it became obvious that a simpler model was needed to provide QoS in the Internet. In August 1997 several presentations related to different service models were made in IETF's IntServ working group session. [Kilkki99 p. 65]

The Differentiated Services working group was established in February 1998. The description of the working group emphasises the need for simple, but versatile methods of providing service differentiation. The DiffServ approach defines a small set of building blocks that will be used and then builds the services from these blocks. [IETF-DiffServ]

3.6.1 DIFFSERV ARCHITECTURE

The DiffServ architecture has benefited from the experiences of the Integrated Services architecture. According to the definition of DiffServ architecture, "*This new QoS architecture should work with existing applications, should not depend on application signalling and should avoid per-microflow state within core nodes*"[BB+98]. Whereas IntServ and RSVP were defined merely for real-time multicast applications, DiffServ is suitable for all kinds of multicast applications, e.g. web browsing and IP phone. [BB+98]

The main concern of DiffServ is scalability and how it can be maintained in future high-speed networks. Scalability is achieved by aggregating the traffic Classification State. Core nodes maintain only a few states in which the packets are classified by using the IP-layer packet marking that employs the Type of Service (TOS) [Almquist92] field. The TOS field is redefined as Differentiated services, DS, [NB+98] field in DiffServ architecture (section 3.6.2). The network nodes employ forwarding behaviour to the

traffic aggregates only. In addition, the sophisticated classification, marking, policing and shaping operations are implemented only at network boundary nodes, where the traffic volume is lower than in the core network.

DiffServ architecture consists of two sets of functional elements: *Boundary* and *Interior* nodes. Boundary nodes classify and handle traffic condition functions, while interior nodes only forward packets based on a DS field. The basic principle of DiffServ is to keep the architecture as simple as possible and therefore it provides service differentiation in one direction only. Unlike RSVP, DiffServ is sender orientated, which means that the traffic sender is responsible for the QoS of the transmission. The other main difference between DiffServ and IntServ architectures is that IntServ offers service classes, while DiffServ specifies the building blocks from which the services can be built. [BB+98]

The nature of IntServ QoS is quantitative, i.e. bandwidth and delays are controlled by the receiver. DiffServ can also be characterised in quantitative terms, but basically it is based on relative priority of access to network resources. The IPv4 precedence marking, as defined in [JN98], is one example of this kind of relative priority marking model.

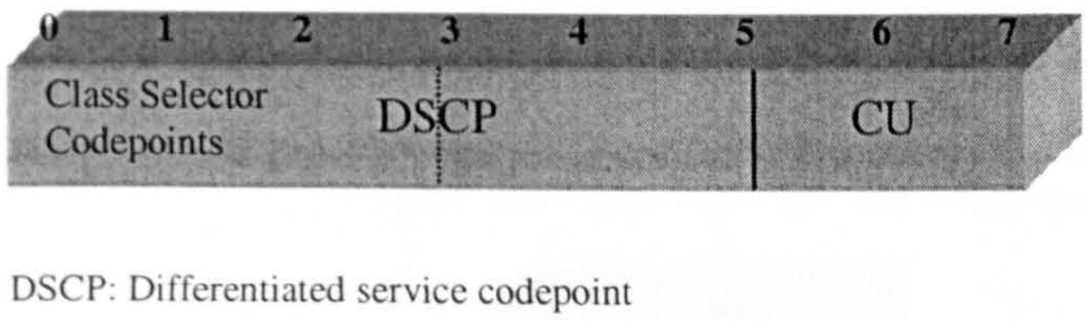
[IK+00] has specified: "*One of the most important benefits of DiffServ is that it takes care of the fragmentation of the Internet, since the Internet consists of several separately administrative domains. When a user, usually a domain, wants to receive DiffServ services, it must have a Service Level Agreement (SLA) with its Internet service provider. SLA is a contract between a user and a service provider, which specifies the DiffServ forwarding characters [BB+98]. DiffServ consists of two main building blocks: the DS field and the per-hop behavior (PHB) [Etal99]*".

3.6.2 DS FIELD

The one of most important building blocks in DiffServ is DS field. DS field is defined in the IP packet's header, where it is intended to supersede the existing definitions of the IPv4 TOS octet [JN98] and the IPv6 Traffic Class octet [Durand01].

Six bits of the DS field are used as a DS codepoint (DSCP). This codepoint is the key input for mapping the packet into a PHB. This mapping is done at each of the nodes along the packet's path.

In addition to the DSCP, the DS field has currently unused (CU) two bits which are reserved for future use and ignored when mapping the packet to a PHB. Figure (3.6) shows the description of the DS field octet. [NB+98]



DSCP: Differentiated service codepoint

CU: Currently unused

Figure (3.6): Description of the DS field

3.6.3 PER-HOP BEHAVIORS

The other main building block of DiffServ is Per-Hop Behaviour (PHB). [Etal99] states, "*A per-hop behaviour is a description of the externally observable forwarding behaviour of a DS node applied to a particular DS behaviour aggregate*". It is important to note that PHB is not a service, but reasonable services can be built by implementing similar PHBs in every node along the traffic path.

As a technical term PHB is a combination of forwarding, classification, scheduling and drop behaviour at each hop. Moreover, PHB is a term that service providers,

administrators and vendors can use for practical discussion. PHBs may be specified in terms of their resource (e.g. bandwidth) priority relative to other PHBs. [Kilkki99 p. 73]. Table (3.5) describes the components of per-hop behaviours.

COMPONENTS	DESCRIPTION
Class Selector (CS)	Maintain compatibility with IP Precedence field (RFC791)
Expedited forwarding (EF)	Can be used to build similar guaranteed service defined in IntServ, which offers guaranteed delay and bandwidth. (RFC2638)
Assured forwarding (AF)	Offers different levels of forwarding assurances. (RFC2597)

Table (3.5): Per-hop behaviour components

3.6.4 DIFFSERV TRAFFIC COMPONENTS

DiffServ includes three traffic components: Traffic classifiers, Traffic profiles, and Traffic conditioners. Figure (3.7) shows the logical view of the DiffServ traffic components. [NB+98]

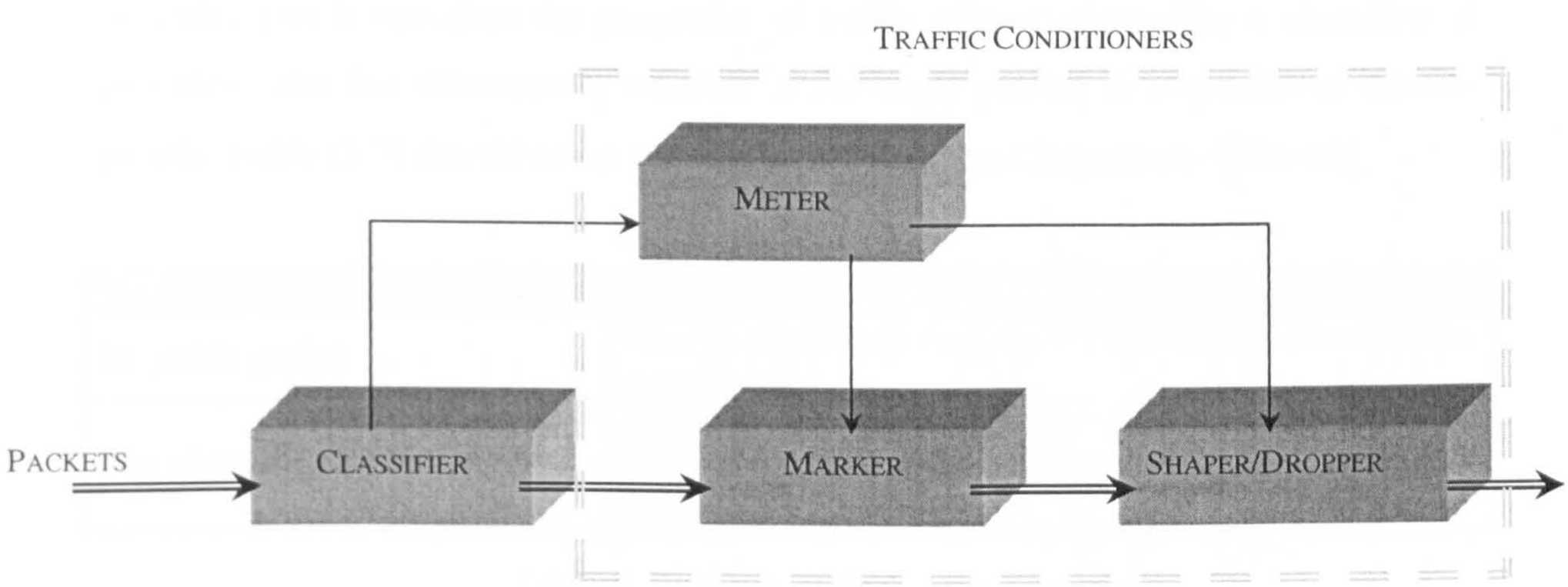


Figure (3.7): Logical view of the DiffServ traffic

- **TRAFFIC CLASSIFIERS**

The Packet classification identifies the traffic, which may receive a differentiated service by being conditioned and/or mapped to one or more behavior aggregates with the DS codepoint. Packet classifiers select packets in a traffic stream based on the content of some portion of the packet header. Table (3.6) shows two types of traffic classifiers. [NB+98]

TRAFFIC CLASSIFIERS	DESCRIPTION
Behavior Aggregate (BA)	Classifiers, which classify packets, based on the DS codepoint only. *(* footnote = in this work we use BA classifiers)*
Multi-Field (MF)	Classifiers select packets based on the value of a combination of one or more header fields.

Table (3.6): Types of traffic classifiers

- **TRAFFIC PROFILE**

A traffic profile specifies the properties of traffic stream selected by a classifier. It provides rules for determining whether a particular packet is in-profile or out-of-profile. Table (3.7) describes the two types of Traffic profile packets. [NB+98]

TRAFFIC PROFILE PACKETS	DESCRIPTION
In-profile packet	May be allowed to enter the DS codepoint without further conditioning.
Out-of-profile packet	May be queued until they are in-profile (shaped), discarded (policed), or marked with the new codepoint (re-mark).

Table (3.7): Types of Traffic profile packets

- **TRAFFIC CONDITIONER**

  Table (3.8) shows the traffic conditioner elements.

TRAFFIC CONDITIONER	DESCRIPTION
Meters	Measures the packet that has been selected by a classifier against a traffic profile
Markers	After packet classification, packets marked into premium assured or best effort classes.
Shapers	Delays some or all of packets in a traffic profile. Shapers happen in the edge or boundary nodes, which eliminates jitters.
Droppers	Discards some or all packets in traffic profile.

Table (3.8): Traffic conditioner elements

Note that a traffic conditioner may not necessarily contain all the above elements, for instance, if there was no traffic profile is in effect, packets may only pass through a classifier and a maker. [NB+98]

3.6.5 DIFFSERV RESOURCE ALLOCATION

DiffServ customers can have two types of Service level Specifications (SLSs) [IK+00] between their nodes, dynamic or static. In the case of a static SLS, customer pays e.g. a fixed amount of money every month. Even if the entire bandwidth is not being used all the time, the customer has to pay for the unused bits as well. This fixed bandwidth allocation may be appropriate for a web service provider who wants to make their web-site seem fast or for a virtual-leased-line service. Static SLSs and manually configured boundary nodes are the first step to Differentiated Services.

The dynamic resource allocation can be realised by the agent called Bandwidth Broker (BB). A BB can be a host, a router or a software process at a router. It can be configured with organisational policies to keep track of the current allocation of AF and EF resources. It can also reject the new PHB resource requests due to the lack of resources. Moreover, a BB can be the only instance that can configure the leaf routers. The configuring is done according to the policy database of the BB. This will reduce the heaviness of the management burden inside the network domain. [NJ99]

The use of BB can be divided into two categories: BBs are used only in network configuration, they make admission control decisions and are used also in dynamic resource allocation.

- **DYNAMIC ADMISSION CONTROL**

In a more sophisticated model a BB is used as an admission controller. Here the reasonable way to communicate between a customer and the local domain BB is the use of RSVP, but "out-of-band" signalling is also possible. When a customer wants a guarantee that there is enough capacity for an end-to-end connection, it sends an RSVP message to the BB of the domain requesting for some DSCP or service for its traffic. At first the BB checks if there is enough capacity left for this request inside its own domain. If there is, the BB sends the message to the BB of the next domain, otherwise the customers will get an error message.
The peer BB processes the request in a similar way and this signalling continues until the request message reaches the BB of the receiver's domain. The signalling process is interrupted if there is not enough capacity available and an error message is sent to the sender. If the message reaches the receiver's BB, "an OK message" is sent to the BB of the previous domain and this signalling ends when the sender host gets the OK message from the local BB. Now the sender can be sure that there is enough capacity along the traffic path for the data transmission to succeed. [NJ99]

- **DYNAMIC BOUNDARY NODE CONFIGURATION**

In the most complex system the BBs located along the traffic path are also used to configure necessary boundary routers. The signalling process starts in a way similar to the previous case. RSVP messages can be used in the signalling between BBs.
When the PATH message reaches the BB of the receiver domain, the BB makes an admission control decision and if the request is accepted it configures the

domain's boundary node. Then the BB will generate a RESV message to the previous domain's BB that will further configure its own domain's boundary router. When the RESV message comes to the sender's BB, the first hop router is configured and the RESV message is returned to the sender. Data transmission can be started and there is enough capacity along the path to forward the sender's traffic. [NJ99]

3.6.6 AN ANALYSIS OF DIFFSERV

The Differentiated Services takes into account the structure of the current Internet and its requirements. The architecture is scalable: routers handle only traffic aggregates services are based on a relative differentiation and no hard guarantees are provided. SLSs are made between large entities or domains.

Since the DiffServ does not (yet) specify services, ISPs have a very important role in realisation of DiffServ. The risk is that ISPs implement PHBs differently so that predictable end-to-end services cannot be achieved. In addition, the network provisioning might be a very difficult task, since no signalling is used and nodes have to be pre-configured.

Especially EF might be very problematic if dynamic end-points are allowed. On the other hand, AF provides more elastic possibilities, but the services built from AF can be too ambiguous. When DiffServ services are provided for end users it is quite clear, that technical terms such as AF or EF are not mentioned. It might be difficult to sell these services (assured service) if their clear advantages, e.g., guaranteed bandwidth, cannot be demonstrated.

Although DiffServ is a very simple and robust architecture, it has a drawback: it has a static nature. The Traffic Conditioning Specification (TCSs) is pre-configured to the boundary routers and the interior nodes have to be pre-dimensioned. [IK+00]

3.7 QUEUING MECHANISMS USED IN INTSERV AND DIFFSERV

This section presents the queuing disciplines, which can be used to implement Integrated and Differentiated Service. In a packet switched network, such as the Internet, queuing discipline has a very important role especially when quality of service is concerned. Queuing is the act of storing packets into a place where they are held for subsequent processing. A typical router has one process for each input and output port. The forwarding process between the ports determines the destination interface to which a packet should be passed. Thus the role of the router is to bind the input processes to the forwarding and scheduling processes and then to the output processes. [FH98 p. 55] Table (3.9) describes the queuing mechanisms.

QUEUING MECHANISM	DESCRIPTION
FIFO QUEUING	FIFO (First In, First Out) queuing is considered to be the standard method for store-and forward traffic handling. Packets that enter the input interface queue are placed into the appropriate output interface queue in the order in which they are received.
PRIORITY QUEUING	Priority queuing is considered as a primitive form of traffic differentiation. The priority queuing is based on the concept where certain type of traffic is always transmitted ahead of other type of traffic.
CLASS-BASED QUEUING	Class-Based Queuing (CBQ) or custom queuing is a variation of priority queuing. The CBQ can be configured to forward a specified number of bytes from a queue in each service rotation time. CBQ provides mechanism to allocate dedicated portions of bandwidth to specific types of traffic. The drawback of CBQ is that it requires some amount of computational processing.
WEIGHTED FAIR QUEUING	Weighted Fair Queuing (WFQ) is a flow-based algorithm, which provides fair bandwidth allocation. WFQ is based on the fair queuing scheme, which operates according to the round-robin (ref) principle.
RANDOM EARLY DETECTION	Random Early Detection (RED) provides a mechanism to avoid a total congestion of TCP flows and the problem of global synchronization, which can be caused by simultaneously TCP slow starts. RED operates by monitoring the queue depth. The average queue size is compared to two thresholds, a minimum threshold and a maximum threshold. When the average queue size is less than minimum threshold no packets are marked to be dropped.

Table (3.9): Queuing mechanisms

3.8 INTEROPERATION BETWEEN INTSERV AND DIFFSERV

Both QoS architectures, Integrated and Differentiated services, have their own advantages and disadvantages. As it has mentioned in section (3.4), IntServ together with RSVP signaling provide dynamic admission control, end-to-end predictable services and an efficient use of a network resources. However, the processing overhead in the high-speed networks will be unacceptable. On the other hand, DiffServ in section (3.6) offers a scalable way to provide QoS in large networks. The lack of a dynamic admission control mechanism and difficulties in resource allocation can make the DiffServ quite unpredictable. By combining the advantages from both architectures it might be possible to build a scalable system, which would provide predictable services. [BF+00]

3.8.1 MOTIVATION FOR INTEROPERATION MODEL

As it mentioned already, the lack of explicit admission control may cause the distortion of exciting traffic flows when new flows join to the aggregate and the resources of the aggregate have already been consumed. As a result, none of the sessions will obtain satisfactory service, when in fact there are resources available for most of the sessions.

When the explicit admission control is used, the new flows are blocked and the senders are informed about the network condition. When the sender receives a rejection signal, it might respond by requesting smaller traffic profile. [BF+00]

3.8.2 FRAMEWORK

This framework, which is proposed by IETF to provide end-to-end QoS, consists of IntServ regions at the periphery of the network and DiffServ regions in the core network. Basically IntServ regions are customers for DiffServ regions, which offer transport services.

The DiffServ regions may, but are not required to, participate in end-to-end RSVP signaling for the propose of optimizing resource allocation and supporting admission control. [BF+00]

3.8.3 DIFFSERV AND INTSERV ARCHITECTURES COMPATIBILITY

IntServ and DiffServ have complementary architectures for the pursuit of end-to-end QoS, where IntServ enables hosts to request pre-flow, quantifiable resources along end-to-end data paths and to obtain feedback regarding admissibility of these requests, DiffServ enables scalability across large networks [BF+00]. The benefits of interoperation of IntServ with DiffServ aggregate traffic control can be described as follows [EH+00]:

- **SCALABILITY**

  DiffServ aggregate traffic control has more scalability compared to IntServ for high-speed transmission in large networks.

- **RESOURCE BASED ON DYNAMIC ADMISSION CONTROL**

  IntServ uses an explicit setup mechanism (e.g. RSVP) to request resources, and DiffServ network region may accept or reject these requests in response. With the use of RSVP signaling in a DiffServ network region, all nodes in this network will have dynamic admission control, which helps to assure that network resources are optimally used.

- **ASSISTANCE IN TRAFFIC IDENTIFICATION/CLASSIFICATION**

  With the DiffServ network regions, traffic is allotted service based on the DSCP marked in each packet's IP header. Thus, in order to obtain a particular level of service within the DiffServ network regions, it is necessary to affect the marking of correct DSCP in packet headers.

- **TRAFFIC CONDITIONING**

IntServ components are able to condition traffic at per-flow granularity, by some combination of shaping and/or policing. Pre-conditioning traffic in this manner before it is submitted to the DiffServ region of the network is beneficial. It enhances the ability of DiffServ region of the network to provide quantitative services using aggregate traffic control.

3.8.4 ALTERNATIVE APPROACHES AND THE COMPARISONS

IETF has classified four alternative approaches instead of DiffServ to co-operate with IntServ architecture for an end-to-end QoS. However, the comparisons have shown that the DiffServ architecture is more compatible and easy to support IntServ architecture. [EH+00]

3.8.4.1 RELATIVE PRIORITY MARKING

- **MODEL**

In this model, the application, host, or proxy node selects a relative priority or precedence for a packet (e.g. delay or discard priority), and the network nodes along the transit path apply the appropriate priority forwarding behavior corresponding to the priority value within the packet's header. The example of this model includes IPv4 precedence marking, 802.5 token ring priority [ISO/IEC98], and the default interpretation of 802.1p [ISO/IEC00] traffic classes. [EH+00]

- **COMPARISON**

DiffServ architecture can be considered as refinement to this model, since it is more clearly specify the role and importance of boundary nodes and traffic conditioners, and since our per-hop behavior model permits more general forwarding behaviors than relative delay or discard priority. [EH+00]

3.8.4.2 SERVICE MARKING

- **MODEL**

In this model each packet is marked with a request for a "Type of service", which may include "minimize delay", "maximize throughput", "maximize reliability", or "minimize cast". Network nodes may select routing paths or forwarding behavior, which are suitably engineered to satisfy the service request. This service request is associated with each individual packet, whereas some service semantics may depend on the aggregate forwarding behavior of a sequence of packets. An example of a service-marking model is IPv4 ToS. [Almquist92]

- **COMPARISON**

This model is subtly different from DiffServ architecture, where the TOS marking [Almquist92] are very generic and do not span the range of possible service semantics. This model does not easily accommodate growth in the number and range of future services (since the codepoint space is small) and involves configuration of the "ToS forwarding behavior" association in each core network node. Standardizing service marking implies standardizing service offering, which is outside the scope of IETF. [NB+98]

3.8.4.3 LABEL SWITCHING

- **MODEL**

In this model path forwarding state and traffic management or QoS state is established for traffic streams on each hop along a network path. Traffic aggregates of varying granularity are associated with a label switched path at an ingress node, and packets within each label switched path are marked with a forwarding label that is used to look up the next-hop nod, the per-hop forwarding behavior, and the replacement label at each hop. Examples of this model include Frame Relay, ATM, and MPLS. [RF95]

- **COMPARISON**

This model permits finer granularity resource allocation to traffic streams, since label values are not globally significant but are only significant on a single link; therefore resources can be reserved for the aggregate of packets received on a link with a particular label, and the label switching semantics govern the next-hop selection, allowing a traffic stream to follow a specially engineered path through the network.

This improved granularity comes at the cost of additional management and configuration requirement to establish and maintain the label switch paths.

In addition, the amount of forwarding state maintained at each node scales in proportion to the number of edge nodes of the network in the best case, and it scales in proportion with the square of the number of edge nodes in the worst care, when edge-edge label switched paths with provisioned resources are employed. [ATMFORUM90]

3.8.4.4 STATIC PER-HOP CLASSIFICATION

- **MODEL**

This model eliminates the requirement for hop-by-hop signaling by utilizing only "static" classification and forwarding policies, which are implemented in each node along a network path. These policies are updated on administrative timescales and not in response to the instantaneous mix of microflows active in the network. [BB+98]

- **COMPARISON**

The state requirements for this model are potentially worse than those encountered when RSVP is used, especially in backbone nodes, since the number of static policies that might be applicable at a node over time may be larger than the number of active sender-receiver sessions that might have installed reservation state on a node. Although the support of large numbers of classifier rules and forwarding policies may be computationally feasible, the management burden associated with installing and maintaining these rules on each node within a backbone network which might be traversed by a traffic stream is substantial. [BB+98]

3.8.5 REFERENCE MODEL FOR INTSERV AND DIFFSERV INTEROPERATION

The IETF reference model includes a DiffServ region in the middle of larger network supporting IntServ end-to-end. The DiffServ region contains a mesh of routers, at least, some of which provide aggregate traffic control. The regions outside the DiffServ region (non-DiffServ regions or IntServ regions) contain meshes of routers and attached hosts, some of which support the IntServ architecture. Figure (3.8) illustrates the reference network model in multicast connection. The components of this reference model are defined as follows [BF+00], [EH+00],[BD+01]:

- **DIFFSERV REGION**

  The DiffServ network region supports aggregate traffic control and is assumed not to be capable of MF classification. Depending on the specific realization of the framework, some number of routers within the DiffServ region may be RSVP aware and therefore capable of per-flow signaling and admission control. If devices in the DiffServ region are not RSVP aware, they will pass RSVP messages transparently with negligible performance impact.

- **NON-DIFFSERV REGIONS OR INTSERV REGIONS**

  The network outside of the DiffServ region consists of IntServ capable hosts and other network elements, such as 802, ATM, etc.

- **HOSTS**

  It is assumed that both the sending and receiving hosts use RSVP to communicate the QoS requirements. Also, traffic control in the host may mark the DSCP in the transmitted packets, and shape transmitted traffic to the requirements of the IntServ service in use.

- **END-TO-END RSVP SIGNALING**

It is assumed that RSVP signaling messages travel end-to-end between hosts (Sender and Receivers) to support RSVP/IntServ reservation outside the DiffServ network region.

- **EDGE ROUTERS**

Edge routers reside adjacent to the DiffServ network regions. The functionality of the edge routers varies depending on the specific realization of the framework.

- **BORDER ROUTER**

Border routers reside in the DiffServ network region. The functionality of the boarder routers varies depending on the specific realization of the framework.

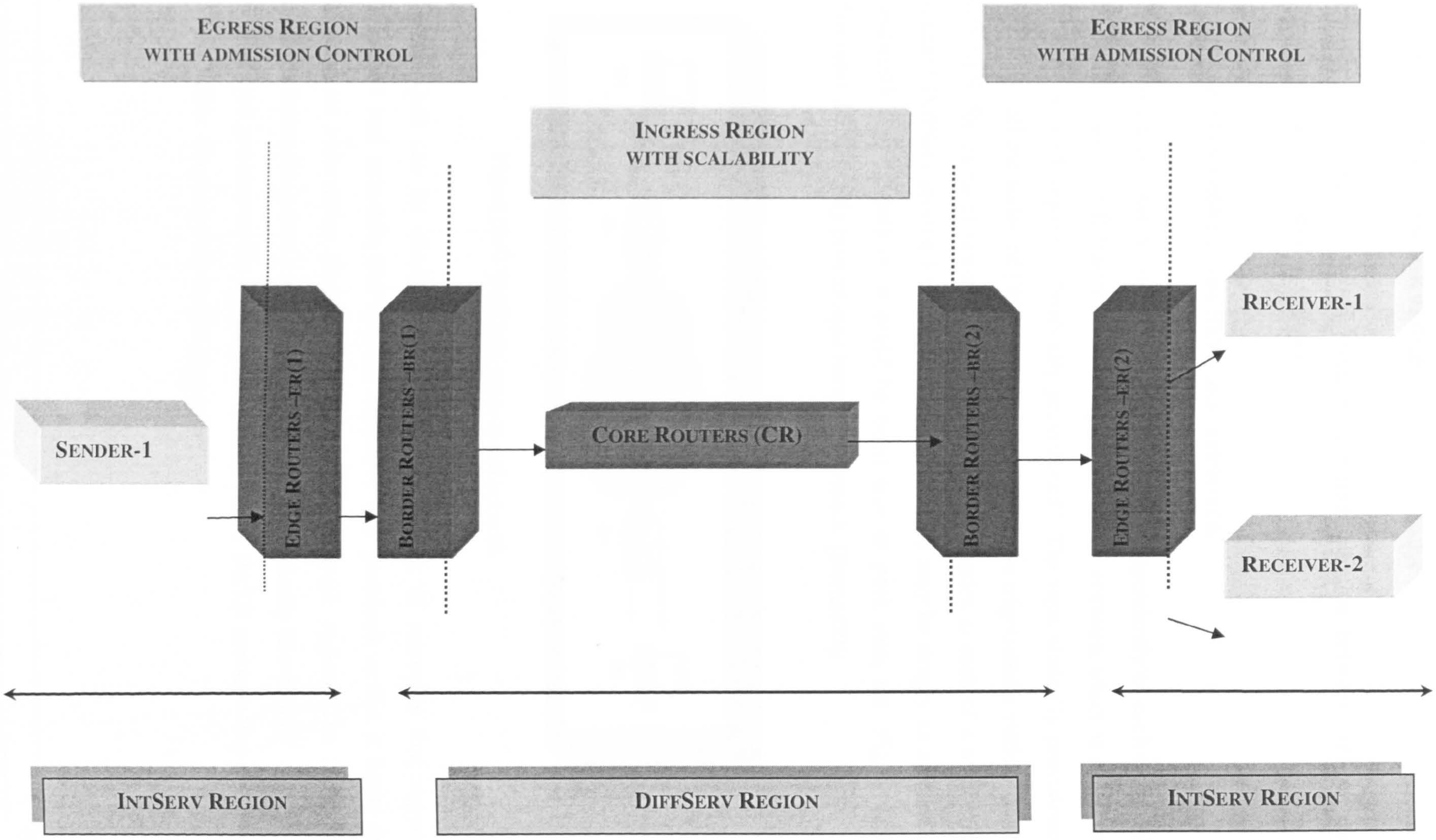


Figure (3.8): Reference model for IntServ and DiffServ interoperation

3.9 APPROACHES ON NETWROK REGION

There have been proposed two approaches for interoperation between IntServ and DiffServ, which are described as follows:

3.9.1 STATICALLY PROVISIONED NETWORK APPROACH

This approach assumes to run IntServ and DiffServ independently of each other. All the nodes (routers) in DiffServ network region are RSVP unaware; which is called the DiffServ network region is "statically provisioned". The term statically provisioned means that all the nodes of DiffServ network region have negotiated a static contract with SLS, for transmit capacity to be provided to the nodes at each of a number of standard DiffServ service levels. The transmit capacity may be simply an amount of bandwidth (throughput) or it could be burst size or peak rate, etc. Figure (3.9) illustrates the statically provisioned network approach. [Bernet00]:

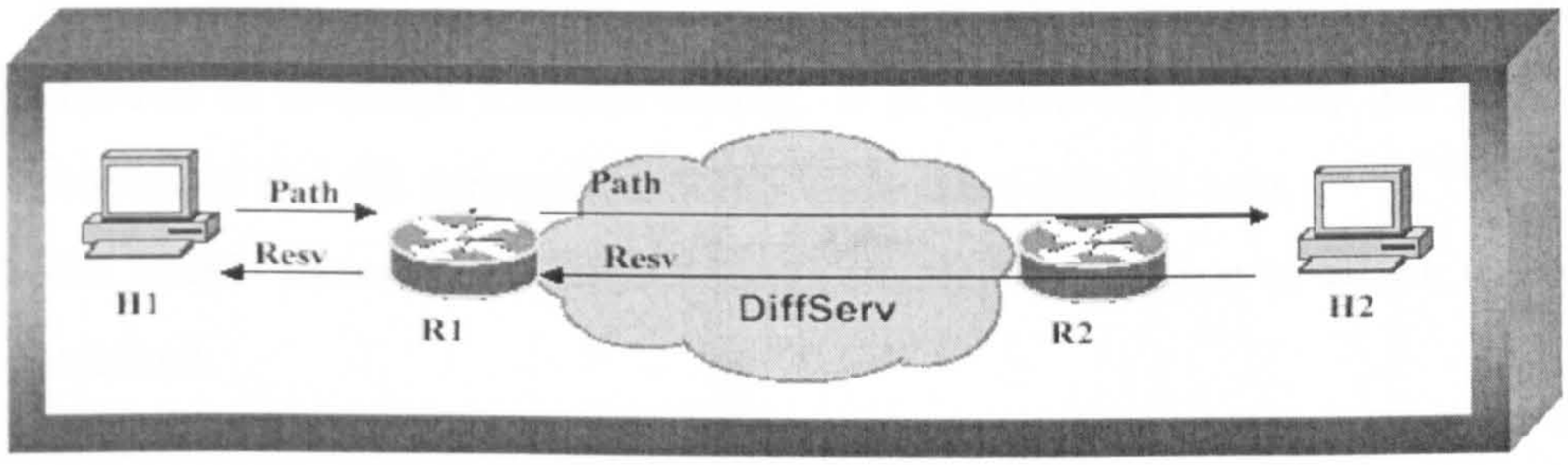


Figure (3.9): Statically provisioned network

End-to-end QoS can be obtained through a combination of networks that support RSVP/IntServ and networks that support DiffServ. The drawback is that it limits the RSVP resource reservation throughout the DiffServ network region. The following sequence illustrates the process when RSVP has been set statically [Bernet00]:

1. The QoS process on the sender generates an RSVP PATH message that describes the traffic offered by the sending packets.

2. The PATH massage is carried toward the receivers. In the network region to which the sender is attached, RSVP processing is applied at capable network elements.
3. At the Edge router, the PATH message is subjected to RSVP processing and PATH state is installed in the router. The PATH message is sent onward to the DiffServ network region.
4. The PATH message is ignored by the routers in the DiffServ network region and then processed at the Edge router according to RSVP processing rules.
5. When the PATH message reaches the receivers, the operating system generates an RSVP RESV message, indicating interest in offered traffic of a certain IntServ service type.
6. The RESV message is carried back towards the DiffServ network region and the sender.
7. At edge router (2), the RESV message is subjected to RSVP processing.
8. In edge router (1), the RESV message triggers admission control processing.
9. If edge router (1) approves the request, the RESV message is admitted and is allowed to continue towards sender. If it rejects the request, the RESV is not forwarded and the appropriate RSVP error messages are sent.
10. The RESV message proceeds through the network region to which the sender is attached.
11. At sender, the QoS process receives the RESV message.
12. The sender may mark the DSCP in the headers of packets that are transmitted on the admitted traffic flow.

3.9.2 DYNAMICALLY PROVISIONED NETWORK APPROACH

The pervious model presumes that the DiffServ network is statically provisioned i.e. there is no signaling between the IntServ network regions and the DiffServ network region, (RSVP PATH and RESV messages are transparent for the DiffServ nodes).
It might be difficult to make efficient use of the resources in the DiffServ region, since the admission control does not consider the availability of resources inside the DiffServ region along the specific traffic path.

This approach assumes the existence of have RSVP signaling on both IntServ and DiffServ network regions. Although all the routers in DiffServ network regions are able to participate in some form of RSVP signaling, they classify and schedule traffic in aggregate, based on DSCP, not on the pre-flow classification criteria used by RSVP/IntServ routers. It can be said that their control-plan is RSVP while their data-plan is DiffServ. The benefit of this approach is to exploit the benefits of RSVP signaling while maintaining much of the scalability associated with DiffServ.

By contrast with the previous approach, when the DiffServ network region is RSVP aware, the admission control agent dynamically works as a part of the DiffServ network region. As a result, changes in the capacity available in the DiffServ network region can be indicated to the IntServ-capable nodes outside the DiffServ region via RSVP. By including RSVP signaling on the routers in the DiffServ network region, it improves the efficiency of resource usage within the DiffServ region and the level of service that the resource requested at admission control. Figure (3.10) illustrates dynamically provisioned network approach.

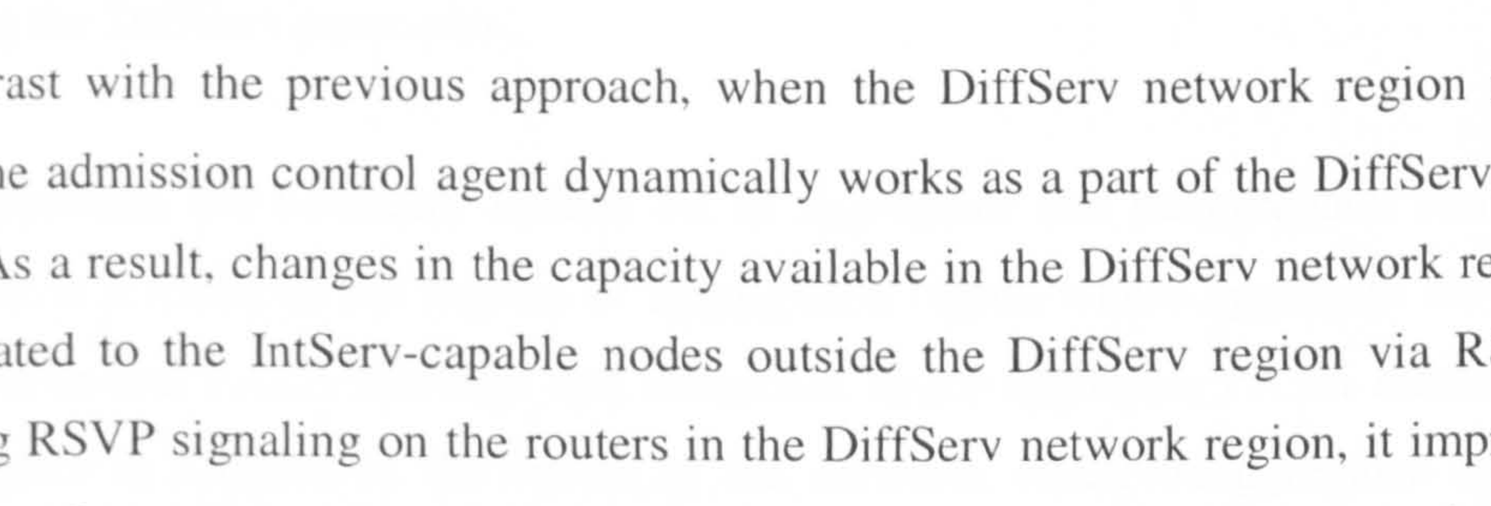


Figure (3.10): Dynamically provisioned network

The use of RSVP signaling referred to as " topology aware admission control". The benefit of supporting RSVP signaling within the DiffServ network region is that it is possible to effect changes in the provisioning of the DiffServ network region (e.g. allocating more or less bandwidth to the EF queue in a router) in response to resource requests from outside of the DiffServ region. Section (3.9.3.1) has been introduced a mechanism that may be used with the DiffServ network region to support dynamic provisioning and topology aware admission control. [QK01], [BF+00]

3.9.2.1 AGGREGATED OR TUNNELED RSVP

This approach is to re-provision the core routers and other border routers using aggregated RSVP to reserve resources between edges of the DiffServ network region and IntServ network regions.

It assumes that the RSVP reservation is mapped into DiffServ PHBs at the edge router. The classification and scheduling states in the core network region are managed according the DiffServ principles.

In this approach, the boundary routers set as aggregator and deaggregator routers. The aggregator is a router at the ingress of aggregation region where aggregate the standard RSVP end-to-end PATH message and composes larger aggregate PATH messages. The Deaggregator is a router at the egress of the network, where deaggregates aggregated PATH messages and consumes aggregate RESV messages.

The aggregate reservations are built between particular aggregator and deaggregator pairs. Those routers have to be known before the aggregate reservation can be done.

By using DiffServ mechanisms for classification and scheduling of traffic supported by aggregate reservations, the amount of classification and scheduling state in the aggregation region is even further reduced. In addition, DiffServ DSCPs are used to identify traffic covered by aggregate reservation. [BI01]

- **END-TO-END GUARANTEED QOS PROCESS**

A RSVP aggregated session is identified by the session object, which includes the IP destination address, the protocol ID and the DSCP information. Diffserv DSCPs are used in DCLASS [BF+00] to identify traffic covered by aggregate reservations in RSVP.

The first router in the transit domain that handles the aggregated reservations is called Aggregator, while the last router in the transit domain that handles the reservations is called Deaggregator. An RSVP aggregation region is consisted by routers that are capable of managing the RSVP aggregated states.

The RSVP Aggregator can use a policy that can be based on local configurations and local QoS management architectures, to set the DSCP packets that are passing into the aggregated region. For example, the Aggregator may be a PSTN (Public Switched Telephone Network) gateway that aggregates a set of incoming calls and makes an aggregate reservation across one or more Diffserv domains up to the Deaggregator that can be e.g., another PSTN gateway. In this situation the call signalling is used to establish the E2E reservations.

When the PATH message finally arrives at a deaggregator, the deaggregator should update its ADSPEC, before forwarding the PATH message towards the receiver. This update is to reflect the impact of the aggregation region onto the QoS to be achieved end-to-end by the flow. Such information can be collected by the ADSPEC of aggregate PATH messages traveling from the aggregator to the deaggregator. Therefore, to enable correct updating of the ADSPEC, a deaggregating router may wait for the arrival of an aggregate PATH before forwarding the end-to-end PATH.

Each endpoint can initiate a per-flow and end to end RSVP resource reservation procedure. The per-flow RSVP resource reservation messages are referred as E2E PATH/RESV and the aggregated resource reservations messages are referred as aggregated PATH/RESV. The E2E PATH/RESV messages are carried through the RSVP aggregation region transparently, since their RSVP protocol Id at the aggregator will be replaced with a new protocol type, i.e., RSVP-E2E-IGNORE.

Each E2E RSVP resource reservation request that arrive at the ingress router, i.e., Aggregator, of an RSVP aggregated region can initiate or resize a RSVP aggregated reservation. In the RSVP aggregated region the aggregated RSVP messages are managing the reservation of the resources for a set of per-flow RSVP reservations. The E2E RSVP reservation states are temporary states, i.e. soft states that have to be updated regularly. This means that E2E PATH and E2E RESV messages will have to be periodically retransmitted. If these states are not refreshed then they will be removed. These states may also be removed by using the E2E PATHTear and E2E RESVTear messages. When these E2E states are removed then their FLOWSPEC information must be removed from the allocated portion of the aggregate reservation, such that this same bandwidth will be re-used for other traffic in the near future. Furthermore, their Sender_TSpec information must also be removed from the aggregated state.

This can be achieved by using some level of trend analysis. From the above information we can deduce that based on a certain policy the Aggregator and Deaggregator will decide when the RSVP Aggregated states will be refreshed or updated and therefore this triggering time is not completely defined by the E2E RSVP messages. The *advantage* of this approach is that it offers dynamic, topology aware admission control to the DiffServ network region without requiring the level of RSVP signaling processing that would be required to support per-flow RSVP. [QK01]

3.10 ANALYZE OF INTEROPERATION OF INTSERV AND DIFFSERV

Section (3.8) has discussed the possibilities of how the IntServ/RSVP principles can be used in parallel with DiffServ principles. Since DiffServ is basically based on the Static network configuration and admission control, the use of RSVP signaling can bring a lot of flexibly into DiffServ architecture. The signaling also enables more accurate and dynamic policing, since the admission control may be based on the policing of a sender, a receiver or an application.

The overall benefit of the combined architectures is the end-to-end services. The RSVP signaling provides knowledge about availability of resources along the traffic path. In this way predictable end-to-end services can be guaranteed. Usage of DiffServ principles eliminates the need of per-flow states along the traffic path. However, such an approach raises some problems, and therefore cannot be an only solution to provide end-to-end QoS for multicast application requirements. They can be described as follow [AS01]:

- Within larger networks the resource management might become impossible. A distribution of different resources based on the RSVP aggregated or per-flow is complex, because network reconfigurations might interfere.

- Enabling the DiffServ network region with dynamic admission control using aggregated RSVP has some drawbacks. Using aggregated RSVP resource management in the DiffServ network region is most likely to be feasible only within a single administrative domain, as each domain will probably choose its own mechanism to manage its resources. This leads to increase complexity and the need of predictive management of bandwidth.

- To cope with the scalability and dynamic admission control behaviours, it is more convenient to propose an independent approach for dynamic admission control. Several activities could be solved locally and a core network only queried, if absolutely necessary.

3.11 SUMMARY

This chapter presented an overview of QoS management operations, parameters and services. QoS management is defined as a concrete realization of required levels of QoS in real system, which is described in three components: negotiation, maintenance and monitoring.

QoS services are described in two services: Integrated services (IntServ) and Differentiated services (DiffServ). IntServ architecture is designed to provide end-to-end resources and dynamic admission control over network. DiffServ architecture is designed to provide scalability over network.

The Interoperation of IntServ and DiffServ by IETF showed that it could achieve end-to-end resource, dynamic admission control and scalability over network providing end-to-end QoS for multicast applications. The next chapter proposes new components to enable the ECTP tree-based transport protocol with QoS management (ECTP QoS management specification) operations using IntServ and DiffServ principles. ECTP transport protocol itself cannot guarantee the QoS levels required by applications. Therefore to enable QoS management functionality in ECTP, the new functions and elements needed to be added on the ECTP connection lifetime.

CHAPTER 4

QOS MANAGEMENT IN ECTP

4. OVERVIEW

As it mentioned in Chapter 2, section (2.6.2), the tree-based multicast transport protocol has been shown the best performance for reliability and scalability (over nodes) compared to other approaches. In Chapter 3, sections (3.4) and (3.6) have mentioned the use of IntServ and DiffServ principles. Also it has discussed in section (3.8), that interoperation between the two architectures can provide an end-to-end resources, dynamic admission control and scalability over the network.

However these two individual solutions (tree-based multicast transport protocol and Interoperation of IntServ and DiffServ) have several drawbacks in the real network. To overcome these problems required a framework, which enables a multicast transport protocol with IntServ and DiffServ functionality. Therefore, multicast transport protocol can provide reliability and scalability (over nodes) along with a dynamic admission control, efficient network resources and scalability over networks. This chapter proposes new components to enable the ECTP transport protocol as a tree-based multicast protocol with QoS management operations using IntServ and DiffServ principles.

4.1 ECTP QOS MANAGEMENT

ECTP transport protocol cannot itself guarantee the QoS levels required by applications. Therefore to enable QoS management functionality in ECTP, the new functions and elements should be added on the ECTP connection lifetime.

As has been discussed in section (2.7), ECTP specifications have been designed in two parts. The control part has been defined in section (2.7.1), which specifies the protocol operations for connection establishment and scalability by using the tree-based approach.

The second part is QoS management specification, which describes QoS management functions for the stable management of QoS for users in a simplex connection. This functionality enables ECTP to reserve resources to support multicast applications throughout of the connection. The ECTP transport protocol supports QoS management through three operations: negotiation, monitoring and maintenance.

During the ECTP connection lifetime, QoS negotiation is performed in the connection creations phase (section 2.7.2.1), where the sender proposes the desired target values for each QoS parameters required by applications to all receivers by multicast.
Based on the QoS parameter values proposed by the sender, each receiver might trigger the reservation of the network resources using reservation set-up protocol (section 3.5).

If QoS negotiation is enabled, each receiver can propose modifications to the sender's proposed parameter values. These modified values are based on the negotiated system capacities at the receiver side and network environment. The parameter target values modified by receivers are delivered to sender via ACK messages.

After an ECTP connection is created, the QoS monitoring is activated in the data transmission phase (section 2.7.2), where each receiver is required to measure the parameter values. The measured QoS values are reported to the sender via an ACK packet where it aggregates the parameter status values from the receivers.

Based on the aggregated QoS values, the sender takes QoS maintenance actions to maintain the connection status at a desired QoS level. Specific rules are pre-configured to trigger QoS maintenance actions such as data transmission rate adjustment, connection pause and resume troublemaker ejection and connection termination. Figure (4.1) illustrates the QoS management functionality along with ECTP connection lifetime phases. [KK+02]

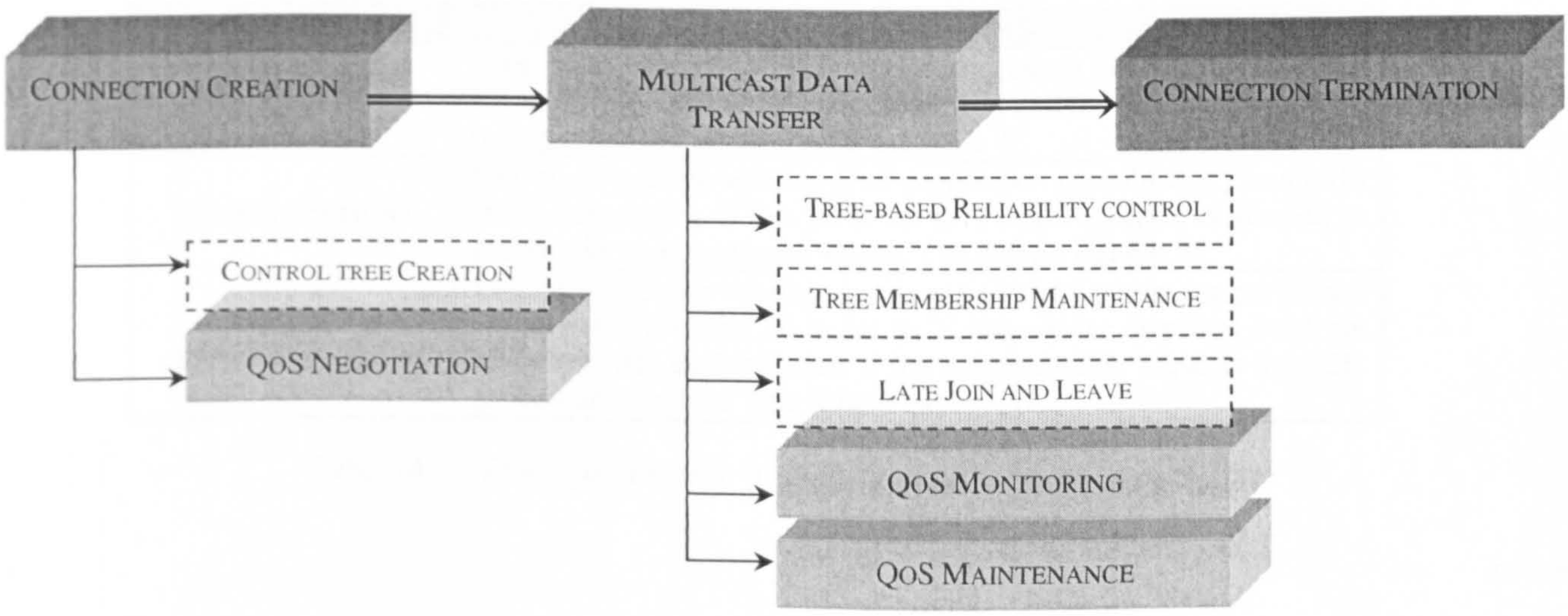


Figure (4.1): ECTP Connection Lifetime

4.2 ECTP QOS MANAGEMENT COMPONENTS

The ECTP protocol components required for QoS management operations could be described as follows (All the components are extended from those already defined in ECTP control specification (section 2.7.2)). [Koh02]

4.2.1 ELEMENTS

The following are the new elements [PK02], [Koh02]:

- **CONNECTION INFORMATION ELEMENT**

  Figure (4.2) shows the connection information element.

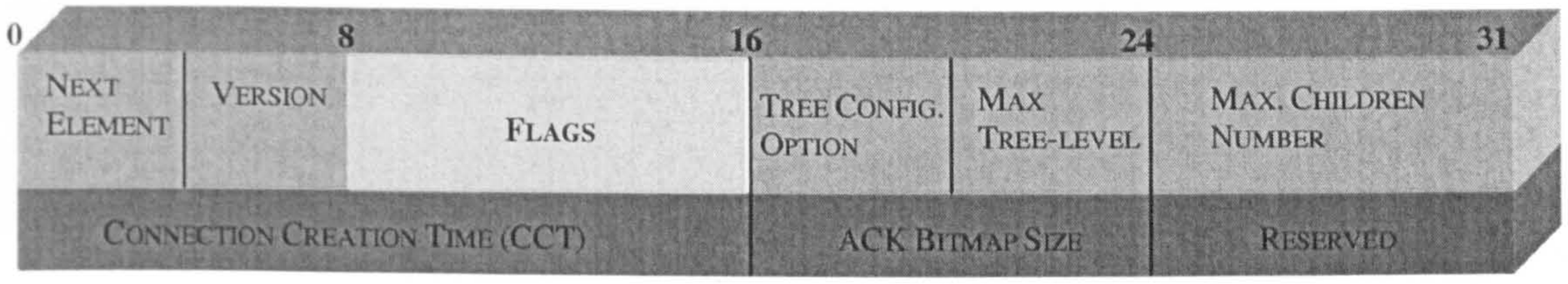
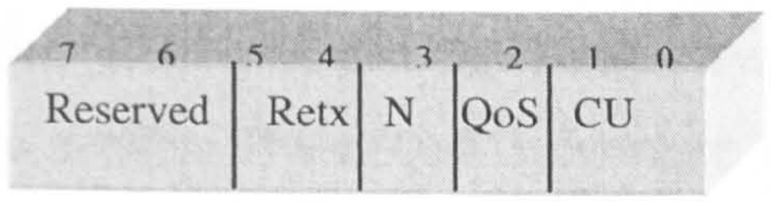


Figure (4.2)- Connection Information Element

Table (4.1) shows that for QoS management, the ECTP sender specifies the following three fields in the 'Flags' byte.



VALUES	DESCRIPTION
QOS	flag bit to indicate whether *QoS management* is enabled (1) or not (0) in the connection. If this bit is set to '1', all the procedures for QoS management are invoked. The default is '0'.
N (NEGOTIATION)	flag bit to indicate whether *QoS negotiation* is enabled (1) or not (0) in the connection. If this bit is set to '1', each receiver is allowed to propose its own parameter values. The default value is '0'.
RETX (RETRANSMISSION)	flag bit to indicate whether *retransmissions* by parent are performed (0) or not (1). If this bit is set to '1', the sender or parent need not retransmit RD packets, even if the retransmission requests by ACK packets are received. The default value is '0'.

Table (4.1): Flag specification in ECTP sender

- **QOS EXTENSION ELEMENT**

A 28-byte QoS extension element specifies the Maximum Segment Size (MSS) and the target values for ECTP QoS parameters. Figure (4.3) shows QoS extension element.

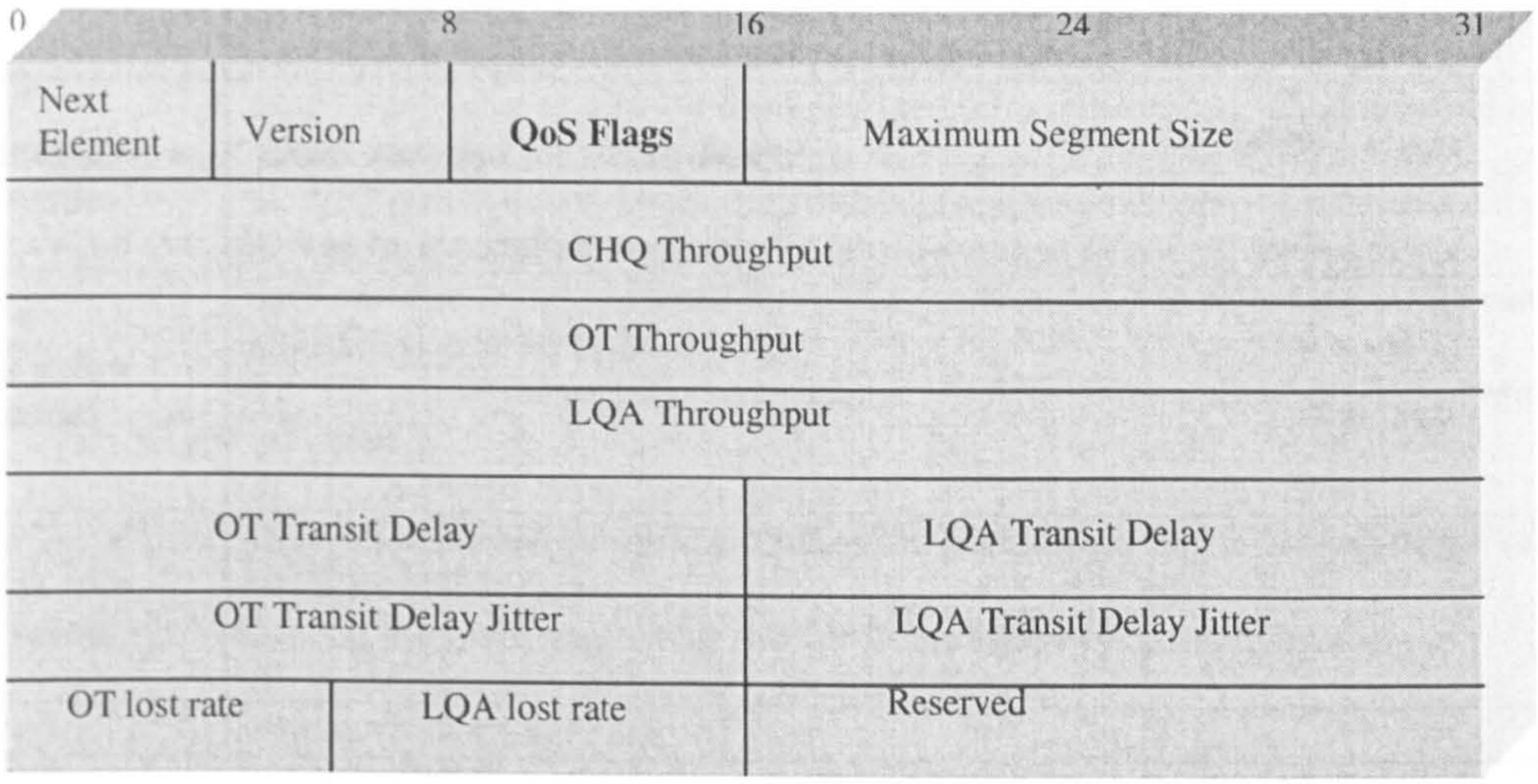


Figure (4.3)- QoS Extension Element

The QoS element is used by ECTP sender to inform the receivers about the target values for QoS parameters by a CR packet in the connection creation phase. In QoS negotiation, the QoS element is also used when a receiver proposes its own modified values to the sender. These QoS values are also referred to in QoS monitoring and maintenance operation. Table (4.2) shows the parameters that specified in QoS extension element.

VALUES	DESCRIPTION
Next element	indicates the type of next element immediately following this QoS element.
Version	defines the current version of this element
QoS flags	is a flag byte to specify if each of QoS parameters and MSS are used in the connection.
Maximum Segment Size (MSS)	Represents the maximum size of ECTP segment or packet in unit of bytes.
Throughput target values	each target value is 32-bit unsigned integer in byte per second. The following target values are valid for throughput: 1. CHQ throughput- upper limit for throughput 2. OT throughput- target throughput for desired display of multicast data 3. LQA throughput-lower limit for throughput
Transit delay target values	each target value is a 16-bit unsigned integer in millisecond. The following target values are valid for transit delay: 1. OT transit delay- target transit delay for desired display of multicast data 2. LQA transit delay-maximally allowed transit delay
Transit delay jitter values	each value is 16-bit unsigned integer in millisecond. The following target values are valid for transit delay jitter: 1. OT transit delay jitter- target transit delay jitter for desired display of multicast data 2. LQA transit delay jitter- maximally allowed transit delay jitter
Data loss rate target values	each value is an 8-bit unsigned integer in percentage. The following target values are valid for data loss rate: 1. OT loss rate- target loss rate for desired display of multicast data 2. LQA loss rate- maximally allowed loss rate
Reserved	is reserved for future use

Table (4.2): Parameters in QoS extension element

- **ACKNOWLEDGEMENT ELEMENT**

In QoS monitoring, each receiver is required to measure the parameter values that have been experienced. A measured parameter value is mapped to a *parameter status* value. A status value is an integer (0,1,2,or 3) where a larger status value indicates a worse status for the connection.

The status values are delivered to sender via ACK packets. The acknowledgement element (section 2.7.5) of the ACK packet contains the status values for QoS parameters used in connection. Figure (4.4) shows the added parameter status in the acknowledgement element.

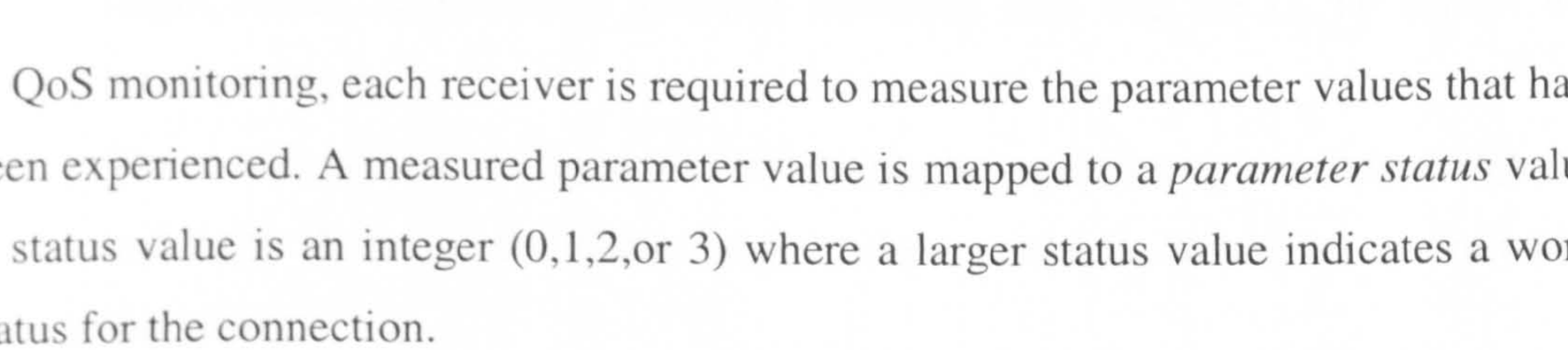


Figure (4.4)-Acknowledgement Element

The following structures are the 8-bit *parameter status*:

1. Two bits indicates the status value for measuring throughput
2. Two bits indicates the status value for measuring transit delay
3. Two bits indicates the status value for measuring transit delay jitter
4. Two bits indicates the status value for measuring data loss rate

4.2.2 QOS PARAMETERS TARGET VALUES

ECTP specification [Koh02] has defined three target values for which may be used by ECTP QoS parameters: *Controlled highest quality* (CHQ), *Lowest quality allowed* (LQA) and *Operating target* (OT). CHQ and LQA values are proposed by the sender and receivers and must be modified their values within this rang. OT value is proposed by the sender and must not be modified by the receivers. The ECTP sender proposes the desired target values for each QoS parameter to all receivers, and all receivers send back their modified target values via ACK messages. The target values can be described as follow[Koh02]:

- **THROUGHPUT (BYTE PER SECOND)**

  As it discussed in section (2.4.3.1), throughput represents an amount of application data output over a specific time period. Based on the throughput target values, applications generate multicast data and the ECTP sender will transmit them.

  The sender should configure the following throughput target values: *CHQ, OT*, and *LQA* throughput. These target values could be described as following inequality:

  $LQAthroughput \leq OTthroughput \leq CHQthroughput$

- **TRANSIT DELAY (MILLISECOND)**

  As it discussed in section (2.4.3.1), transit delay represents an end-to-end transmission time from a sender to a receiver. For desirable display of multicast data, the sender may configure the following target values: *OT* and *LQA* transit delay. These target values could be described using following inequality:

  $OTtransitdelay \leq LQAtransitdelay$

- **TRANSIT DELAY JITTER (MILLISECOND)**

  As discussed in section (2.4.3.1), transit delay jitter represents sudden variations of transit delay values. For desirable display of multicast data, the sender may configure the following target values: *OT* and *LQA* transit delay jitter. These target values could be described in terms of following inequality:

  $OTtransitdelayjitter \leq LQAtransitdelayjitter$

- **DATA LOST RATE (PERCENTAGE)**

  As it mentioned section (2.4.3.1), data lost rate is defined as ratio of amount of lost data over the amount of totally transmitted data. For desirable display of multicast data, the sender may configure the following target values: *OT* and *LQA* data lost rate. These target values could be described using following inequality:

  $OTdatalostrate \leq LQAdatalostrate$

Table (4.3) summarises the relationship between QoS parameters and their target values.

ECTP QoS PARAMETERS	TARGET VALUES	DESCRIPTION
Throughput	CHQ Throughput LQA Throughput OT Throughput	LQA Throughput<=OT Throughput <=CHQ Throughput
Transit delay	LQA Transit delay OT Transit delay	OT Transit delay <=LQA Transit delay
Transit delay jitter	LQA Transit delay jitter OT Transit delay jitter	OT Transit delay jitter <=LQA Transit delay jitter
Data loss	LQA Data loss OT Data loss	OT Data loss <=LQA Data loss

Table (4.3): Relationship between QoS parameters and their target values

4.3 QOS MANAGEMENT PACKETS

Table (4.4) shows ECTP packets that are used in QoS management operation.

	EXTENSION ELEMENT				
PACKET TYPES	CONNECTION INFORMATION	TREE MEMBERSHIP	ACKNOWLEDGEMENT	TIMESTAMP	QOS
CR	✗				✗
CC		✗			✗
IIB		✗		✗	✗
ACK		✗	✗	✗	
JC	✗				✗

Table (4.4)- ECTP packets used in QoS management

The CR packet contains a QoS element. This is used by the sender to propose the target values of QoS parameters that are used in the connection. If QoS negotiation is enabled, each receiver responds to the sender with its own proposed values for QoS parameters, via the CC packet. The sender will arbitrate the returned proposals, and the arbitrated values for the QoS parameters will be delivered to the receiver via HB packets. For a late-joiner the target QoS parameter values currently being used in the connection are notified via JC packet.

The target or negotiated values will be referred in QoS monitoring and maintenance operations. ACK packets are used to convey the status values for QoS parameters experienced at the receiver side. [Koh02]

4.4 ECTP QOS MANAGEMENT PROCEDURES

ECTP QoS management operation can be described as follows [KP+00]:

- QoS negotiation
- QoS monitoring
- QoS maintenance

The QoS negotiation is performed in the connection creation phase, while the QoS monitoring and maintenance are performed in data transmission phase. Table (4.5) shows relationships of the ECTP QoS management operations with ECTP connection phases.

PHASES / OPERATIONS	CONNECTION CREATION	DATA TRANSMISSION	CONNECTION TERMINATION
Negotiation	✖		
Monitoring		✖	
Maintenance		✖	

Table (4.5)- Relationships between ECTP QoS management operations and ECTP connection phases

4.4.1 QOS NEGOTIATION

The QoS negotiation is performed in the connection creation phase when ECTP sender transmits a CR packet to all receivers to start connection creation. The CR packet contains the target values for each QoS parameter such as *CHQ*, *OT* and *LQA*. The sender is required to specify via QoS extension element whether each QoS parameter is subject to negotiation. Before the sender transmits a CR packet, it invokes RSVP signalling protocol to negotiate resource reservation with the receivers.

When QoS negotiation is enabled in the connection, each active receiver responds to the sender with a CC packet that contains the modified target values for the relevant QoS parameters. QoS negotiation will not be performed for late-joining receivers. The ECTP sender just notifies the negotiated parameter values to the late-joining receiver via the JC packet. The negotiated target values would be used in the both QoS monitoring and maintenance. [KP+00]

- **NEGOTIATION PROCEDURES**

The following procedures describe negotiation operations for each QoS parameter and their target values [KP+00]:

1. According to the applications requirements, the sender proposes the target parameter values and transmits them by a CR packet via QoS extension element to all receivers.
2. Before sender transmits CR packets, ECTP invokes the RSVP API to map the target values into RSVP object traffic descriptors, and send the target values via RSVP messages to the receivers for negotiation.

3. According to proposed target values by the sender, each receiver can propose the modified values. During QoS negotiation, receivers must not modify the OT value for each parameter. Finally each receiver delivers the modified values to sender via a CC packet.
4. If a receiver does not wish to modify a QoS parameter, it will just return the same target value that was proposed by the sender.
5. The sender arbitrates the modified parameter values proposed by receivers, which different for each QoS parameters.
6. Sender announces the negotiated parameter values to receivers via HB packets.

4.4.2 QOS MONITORING

The QoS monitoring is performed in the data transmission phase. The purpose of this operation is to provide the sender with the QoS status information during connection. Therefore, each receiver is required to measure the proposed parameter values and reported back to the sender.

For each QoS parameters used in ECTP connection, each receiver measures the parameter values that have been experienced. Then for each parameter the measured value is mapped to a *parameter status* value.

A parameter status value is measured and recorded in the acknowledgement element (section 4.2.1) and it passed on to the sender via an ACK packet. Finally the sender aggregates the parameter status values from all the receivers. Based on the monitored status information, the sender can also take any QoS maintenance actions necessary. [KP+00]

4.4.2.1 QOS PARAMETER VALUES MEASUREMENT

- **THROUGHPUT PARAMETER**

  For the throughput value measurement, a receiver needs to keep information about how many data packets (bytes) have been received for the specific time. Each time a new throughput value is obtained, the value is mapped to the parameter status value. Throughput parameter is calculated as follows [KP+00]:

$$Throughput = \frac{Amountofreceiveddatapakets(bytes)}{AGN\,\mathrm{sec}onds}$$

  where AGN is ACK generation number

- **TRANSIT DELAY AND JITTER**

  For an end-to-end transit delay and jitter measurement, the sender is required to transmit data packet with a timestamp element. Another requirement is the synchronisation of time clocks between sender and receivers.

Without satisfactory resolution of these requirements, it is hard to get exact information about transit delay and jitter. In this work, it is assumed that each receiver can measure an end-to-end transit delay of data packet from the sender.
Transit delay is measured for each data packet received. These transit delay values are averaged over the data packet receiving during AGN seconds.
Transit delay jitter is measured as the difference between the maximum and minimum transit delay values of those received data packets over a defined time period. During the AGN seconds, every time a new data packet arrives, the transit delay is calculated, and the averaged delay and jitter values are updated. Just before a receiver generates an ACK packet, the currently measured value will be mapped to the parameter status value. [KP+00]

- **DATA LOSS RATE**

For the data loss rate value measurement, a receiver needs to keep information about how many data packet have been lost. When the receiver generates an ACK packet, the currently measured data loss rate value is mapped to the parameter status value, which is calculated as follows [KP+00]:

$$Datalostrate = \frac{Numberoflostpakets}{Numberofdatapacketsreceivedduring AGN \sec onds}$$

where AGN is ACK generation number

4.4.2.2 MAPPING TO A PARAMETER STATUS VALUE

The measured parameter value is mapped to a parameter status value for each QoS parameter. The subsequent ACK packet will contain the status value. The measurement of a QoS parameter is activated only if the use of the parameter in the connection is indicated. Figure (4.5) illustrates the mapping functionality from the measured value to a status value. [KC+99]

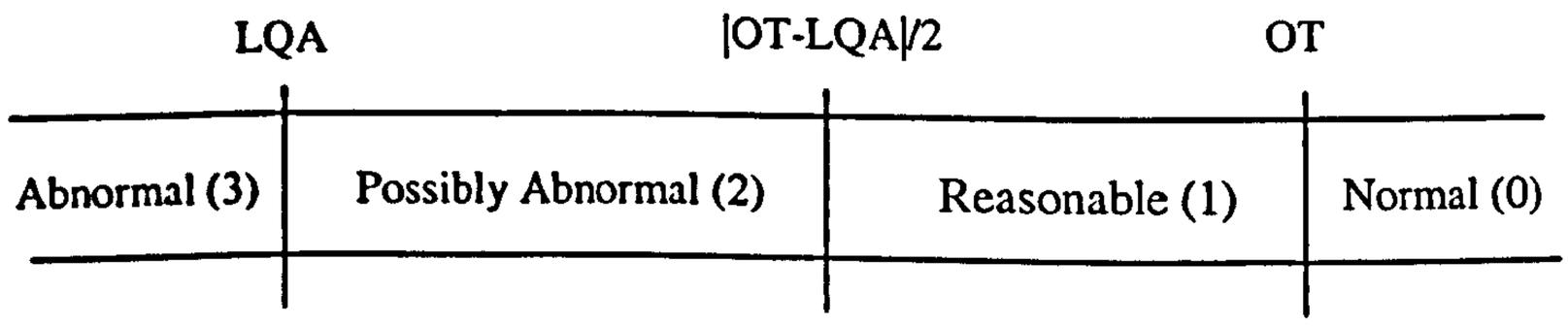


Figure (4.5)- Mapping of Measured value to status value

For this mapping, ECTP uses *threshold* values classify to status values into four categories: *Normal, Reasonable, Possibly Abnormal,* and *Abnormal.* An initial parameter status value is set to '0'. Table (4.6) shows the obtained measured value for the mapping. [Koh02]

TIIRESIIOLD VALUES	PARAMETER STATUS VALUE	MAPPING
NORMAL	0	if "measured parameter value > OT", then Status value =0,
REASONABLE	1	if " threshold <measured parameter value <= OT " then Status value
POSSIBLY ABNORMAL	2	if " LQA < measured parameter value <=threshold", then Status value =2,
ABNORMAL	3	if " measured value <=LQA", then Status =3,

Table (4.6): Obtained measured value of mapping

The threshold value is set to |OT-LQA|/2 as it has been shown in the figure (4.5). These mapping rules are only defined for throughput parameter. For the rest of QoS parameter such as delay, jitter and data loss rate, these rules must be reversed because of OT values≤LQA values. [Koh02]

4.4.2.3 REPORTING TOWARD SENDER

Each receiver reports the mapped parameter status values to its parent via ACK packets. In this fashion, all information about the parameter status value will be delivered toward the sender along the tree hierarchy. Then sender performs aggregation of ACK packets reported from its children every AGN seconds. After aggregation of ACK packets, the sender simply takes an average value weighted by ARN (active receiver number) of each child, which is as follows [KPC+00], [KC+99]. [Koh02]:

$$Aggregated\ status\ value = \frac{Sumofreportedsttauevalues}{ARN}$$

where ARN is Active receiver number.

The sender will have the following aggregated status values for each QoS parameter.

- Aggregated status for throughput is Tvalue.
- Aggregated status for transit delay is Dvalue.
- Aggregated status for transit delay jitter is Jvalue.
- Aggregated status for data loss rate is Lvalue.

The monitored information may be used in the QoS maintenance. Among the monitored values, Lvalue is used for the adjustment of data rate transmission. A weighted sum value for all the status values (Tvalue, Dvalue, Jvalue and Lvalue) can be used to trigger connection pause and termination.

4.4.3 QOS MAINTENANCE

QoS maintenance is performed to maintain the quality of connection at a desired level and to prevent the connection quality from being degraded below the negotiated QoS level. Based on the monitored parameter status values, the sender will take the following actions [KP+00]:

- Adjustment of data transmission rate
- Connection pause and resume
- Connection termination

These actions will be announced to all the receivers via ND and CT packets transmitted by the sender.

4.4.3.1 ADJUSTMENT OF DATA TRANSMISSION RATE

ECTP uses a fixed-sized window-based flow control. The default window size is the same as *ACK bitmap size* (ABS), 32 bytes. Senders can maximally transmit the window size data packets at the rate of Data Transmission Rate (DTR). ECTP performs congestion control by dynamically adjusting DTR, based on the date loss rate status value *Lvalue.* [KPC+00], [Koh+02]

4.4.3.2 CONNECTION PAUSE AND RESUME

- Connection pause can be performed by the sender to suspend temporarily the multicast data transmission, hence to prevent the connection quality from being more severely degraded. [Koh+02]

- Connection resume may be performed according to the request of application. In this case, sending application will trigger the connection pause, based on monitored parameter status values, i.e. *Tvalue*, *Dvalue*, *Jvalue* and *Lvalue*. If the connection pause is triggered, the sender transmits periodic ND packets with the F bit set to '1' in the fixed header. The sender must not transmit any new DT packet, while the control packets including HB packets can be sent. Each receiver may also send control packets such as ACK. [Koh+02]

4.4.3.3 CONNECTION TERMINATION

The connection might be terminated in three different ways [KK+02]:

1. Connection termination phase terminates a connection when all the multicast data have been transmitted.
2. Connection termination may be triggered according to the request of the application where the sender transmits a CT packet to all the receivers and closes the connection
3. Connection termination may be triggered based on the pre-configured connection termination Time (CTT). In this case, connection termination is triggered if a subsequent connection pause occurs again within CTT from connection resume. The CTT timer is activated when connection resume is indicated.

4.5 SUMMARY

This chapter proposed new components to enable the ECTP transport protocol as a tree-based multicast protocol with QoS management operations using IntServ and DiffServ principals. ECTP transport protocol itself cannot guarantee the QoS levels required by applications. Therefore to enable QoS management functionality in ECTP, the new functions and elements should be added on the ECTP connection lifetime. QoS management has three operations: negotiation, monitoring and maintenance. During the ECTP connection lifetime, QoS negotiation is performed in the connection creations phase. After an ECTP connection is created, and if QoS management is enabled, the QoS monitoring is activated. Based on the aggregated QoS values, the sender takes QoS maintenance actions to maintain the connection status at a desired QoS level. Once all the new QoS management components have been defined in ECTP transport protocol, the next chapter proposes two models to map ECTP QoS parameters into the standard and aggregated RSVP resource reservation.

CHAPTER 5

PROPOSED MODELS AND ARCHITECTURES

5. OVERVIEW

As has discussed in Chapter 4, ECTP transport protocol cannot itself guarantee the QoS management over the network. Therefore, new components have been proposed and defined to enable QoS management operations (negotiation, monitoring and maintenance) on ECTP transport protocol.

Once all the QoS management components have been defined for the ECTP transport protocol, the next step is to propose the mapping models to enable ECTP transport protocol to be used with IntServ and/or DiffServ principles for end-to-end QoS management.

As has mentioned in Chapter 3, the RSVP signaling protocol [BZ97] has been designed to provision resources for network traffic. Whereas DiffServ simply marks and prioritizes traffic. RSVP is more complex and demanding than DiffServ in terms of router requirements, so can negatively impact backbone routers. Therefore there has been proposed a framework to use IntServ and DiffServ in parallel to provide an end-to-end QoS. The benefit of this framework is that it should provide dynamic admission control and efficient network resources by IntServ, and scalability over networks by DiffServ.

This chapter proposes two mapping models to enable ECTP transport protocol with end-to-end resource reservation using IntServ and/or DiffServ principles. They are described as follows:

- **MODEL (I)-** *Enabling ECTP transport protocol to negotiate end-to-end resource reservation using the standard RSVP signaling protocol (IntServ).* This model proposes to map ECTP QoS parameters target values into the standard RSVP object traffic descriptors (section 3.5.6). As has been mentioned, the character of the ECTP transport protocol itself is unable to provide QoS management. Therefore an interface between ECTP and RSVP is required so that the ECTP QoS parameter target values would be informed to the RSVP processor. This mapping enables ECTP transport protocol to provide end-to-end connections for multicast applications by negotiating the QoS level in the connection creation phase. QoS negotiation operation uses RSVP signaling to enable ECTP to negotiate the

resource reservation required by applications to all active receivers before transmitting packets.

- **MODEL (II)-** *Enabling ECTP transport protocol to negotiate end-to-end resource reservation using the standard and aggregated RSVP signaling protocol (IntServ and DiffServ).*

  This model proposes to map ECTP QoS parameters target values into the standard and aggregated RSVP object traffic descriptors. This mapping enables the ECTP transport protocol to provide end-to-end connection for multicast applications by negotiating the QoS level in the connection phase. Therefore, ECTP QoS parameter target values are mapped into standard and aggregated RSVP resource reservation to provide end-to-end resources and dynamic admission control as well as scalability over networks.

The proposed models enable the ECTP transport protocol to make resource reservation by negotiation in the connection creation phase before ECTP sends the CR packets to all receivers.

Before discussing these proposed model designs and the corresponding mappings, ECTP and RSVP protocols need to have a specific application-programming interface (API) in order to interact with lower or upper layers. RSVP API (RAPI) [BH98] and ECTP API [Koh02] have been defined by IETF and ITU/ISO respectively. Section (5.1) and (5.2) discuss the RSVP and ECTP APIs.

5.1 RSVP APPLICATION PROGRAMMING INTERFACE

The RSVP application programming interface (RAPI) has been designed by IETF to interact with the multicast applications in order to request required QoS. Figure (5.1) illustrates RAPI's implementation model. [BH98]

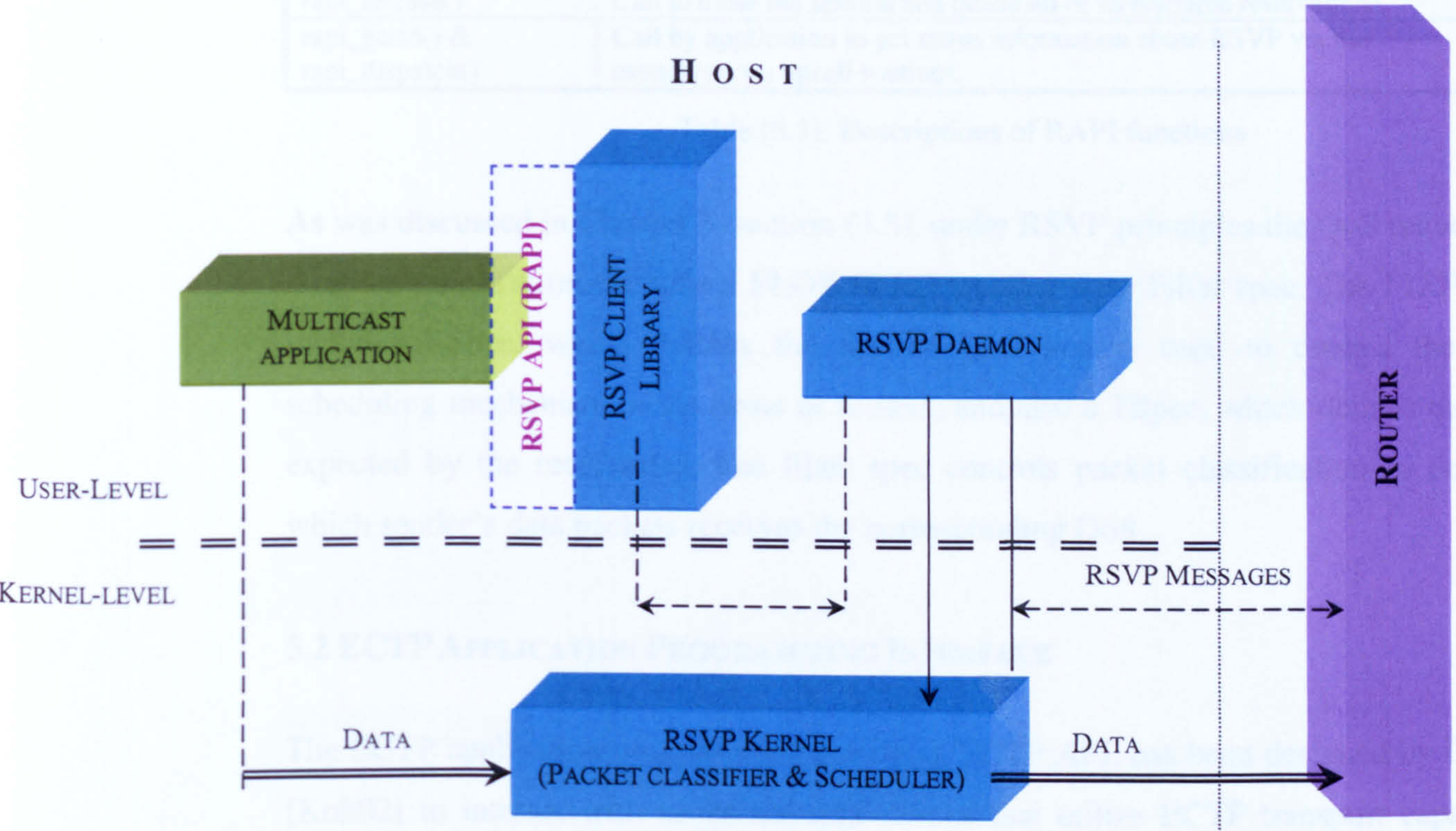


Figure (5.1): RSVP Application programming Interface

These modules can be described in the RSVP implementation model as follows:

- RSVP CLIENT LIBRARY- RAPI is based on a client library module linked with the applications.
- RSVP API- provides an interface with applications and RSVP client library, which ensures the RSVP operations for supporting the applications requirements.
- RSVP KERNEL- manages the data queuing and scheduling according to the requirements from the higher layer RSVP.
- RSVP DAEMON- handles a signaling between RSVP client library and RSVP kernel modules.

Table (5.1) summarizes RAPI routines used by applications. Details of the routines are beyond the scope of this work. [BH98]

RAPI FUNCTION	DESCRIPTION
rapi_session()	Call to define an API session for sending a single simplex data flow and or receiving such data flow.
rapi_sender	Call by a sender to send RSVP PATH message.
rapi_resever()	Call to make a QoS reservation as a data receiver.
rapi_release()	Call to close the session and delete all of its resource reservations.
rapi_getfd() & rapi_dispatch()	Call by application to get status information about RSVP via the asynchronous upcall routines.

Table (5.1): Descriptions of RAPI functions

As was discussed in Chapter 3, section (3.5), under RSVP principles the QoS requested by multicast application contains a FLOWSPEC together with Filter spec. The FLOWSPEC includes RSpec, which defines the desired QoS and is used to control the packet scheduling mechanism in the hosts or routers, and also a TSpec, which defines the traffic expected by the receiver(s). The filter spec controls packet classification to determine which sender's data packets receives the corresponding QoS.

5.2 ECTP APPLICATION PROGRAMMING INTERFACE

The ECTP application programming interface, ECTP API, has been designed by ITU/ISO [Koh02] to interact with multicast applications that utilize ECTP transport capabilities. The API functions have been designed based on the Berkeley socket API functions [Stevens94]. However, to differentiate the ECTP API from the existing Berkeley socket functions, ECTP API functions are named with a prefix "*m*" (e.g. msocket). Table (5.2) summarizes ECTP API functions used by applications. Details of the functions are defined in [Koh02].

ECTP API FUNCTION	DESCRIPTION
msocket()	Creates a new multicast socket in ECTP communication domain.
mbind()	Associates a set of local and group addresses/ports with the socket.
mconnect()	Sender initiates a connection creation to a specified foreign address Late-joiner receiver initiates a join process.
maccept()	Prospective receivers join the ECTP connection by accepting the connection creation signal from the sender.
msend()	Sends application data to a destination group.
mrecv()	Delivers received data to application. Delivers some indication messages for control to application during the data transfer phase.
mclose()	Terminates connection and release socket.
mgetsocketopt()	Gets socket and protocol options from kernel.
msetsocketopt()	Sets socket and protocol options to kernel.

Table (5.2): Descriptions of ECTP API functions

Figure (5.2) illustrates the use of ECTP API functions during the connection lifetime. As shown in Figure (5.2), a receiver takes different actions according to its node type: an early joiner or late-joiner. An early joiner goes into *maccept* mode after *msetsockopt*, while a late-joiner invokes *mconnect* function directly.

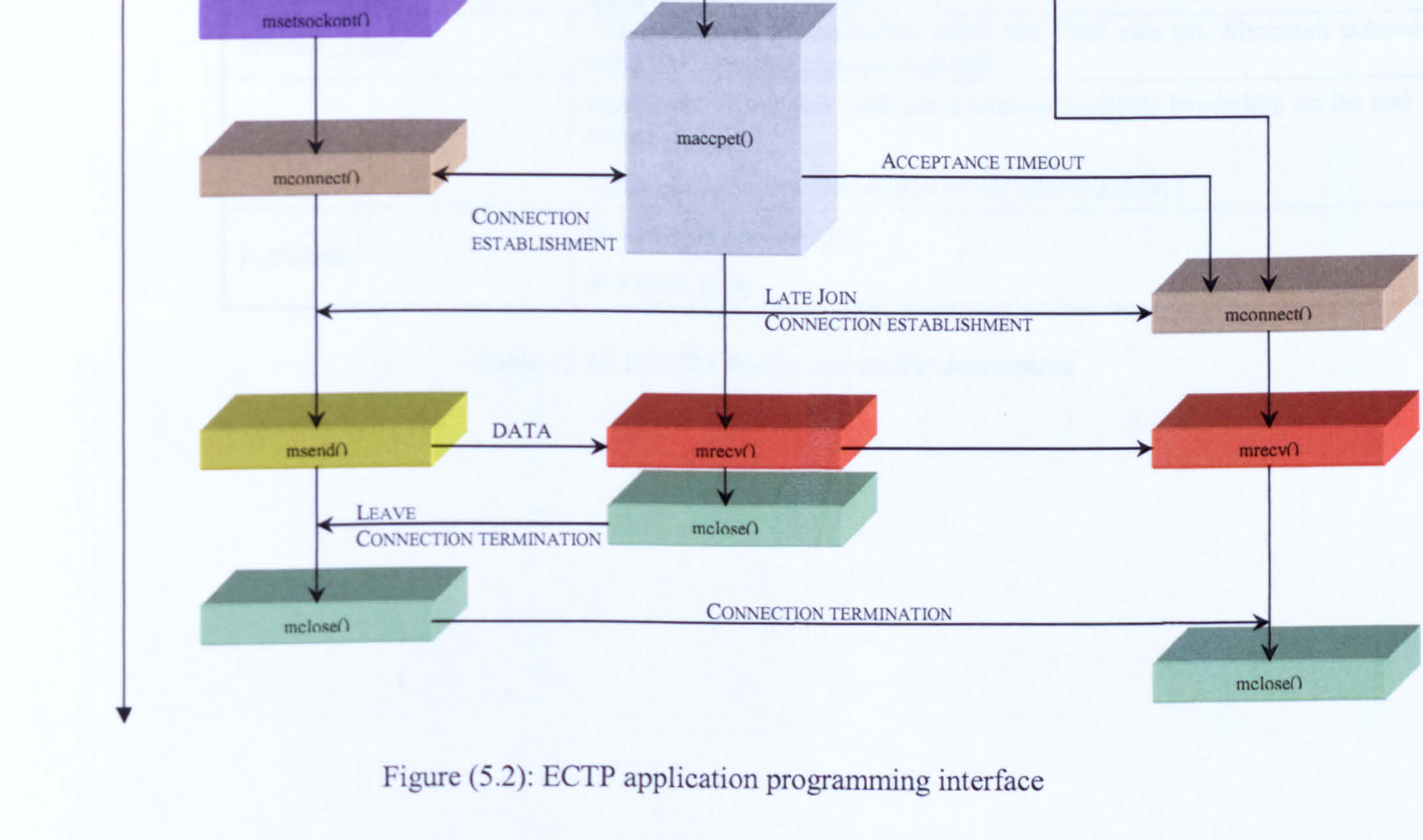


Figure (5.2): ECTP application programming interface

5.3 OVERVIEW OF ECTP QOS PARAMETER AND RSVP OBJECTS

As has mentioned in Chapter 4, section (4.2.2), the ECTP specifications has defined ECTP QoS parameters and their target values. Table (5.3) summarizes the ECTP QoS parameters and their target values.

ECTP QOS PARAMETERS	TARGET VALUES
Throughput	CHQ, OT and LQA
Transit delay	OT and LQA
Transit jitter	OT and LQA
Data loss rate	OT and LQA

Table (5.3): ECTP QoS parameters target values

Also Chapter 3, section (3.5.6) has mentioned that each RSVP messages consist of three objects and within each object there have been defined the traffic descriptors. Table (5.4) summarizes RSVP objects with their corresponding traffic descriptors.

RSVP OBJECT	TRAFFIC DESCRIPTORS
SENDER_TSPEC	Token rate (r), Token-bucket depth (b), Peak rate (p), Minimum policed unite (m), Maximum packet size (M)
ADSPEC	Bandwidth of the link with the minimum available bandwidth on the end-to-end path; End-to-end transmission delays except queuing delay;
FLOWSPEC	R: a reserved bandwidth, S: a slack term.

Table (5.4): RSVP Objects and traffic descriptors

5.4 MODEL (I)- ENABLING ECTP PROTOCOL TO NEGOTIATE END-TO-END RESOURCE RESERVATION USING STANDARD RSVP.

5.4.1 DESCRIPTION

This model proposes to map ECTP QoS parameters target values into the standard RSVP object traffic descriptors (section 3.5). As has mentioned, the character of the ECTP transport protocol itself is unable to provide QoS management therefore an interface between ECTP and RSVP is required so that the ECTP QoS parameter target values would be informed to the RSVP processor. This mapping enables the ECTP transport protocol to provide end-to-end connection for multicast applications by negotiating the QoS level in connection creation phase. QoS negotiation operation (section 4.2) uses RSVP signaling to enable ECTP to negotiate the resource reservation required by applications to all active receivers before transmitting packets. Figure (5.3) illustrates the interworking of ECTP with standard RSVP.

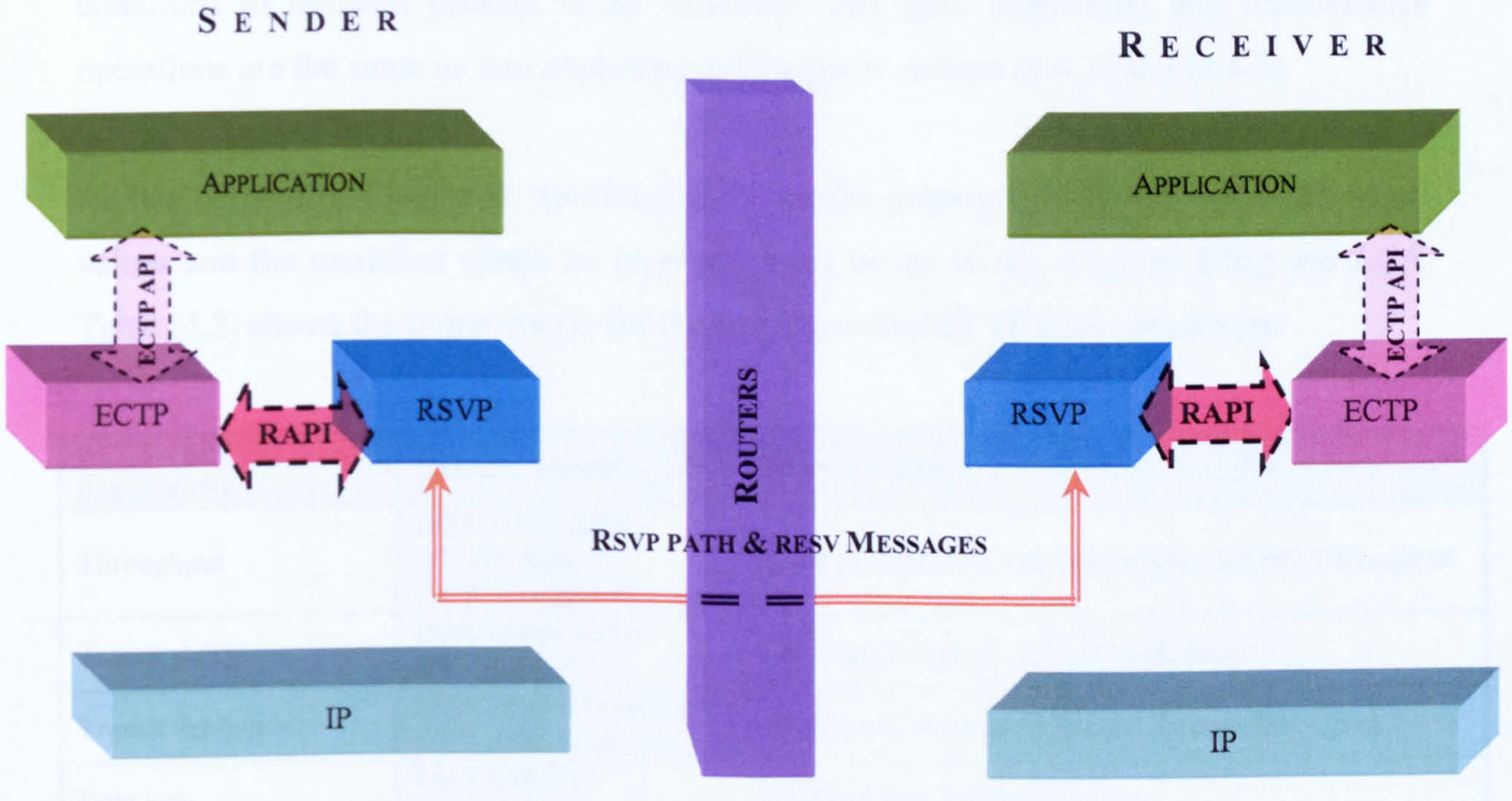


Figure (5.3): Interworking of ECTP with standard RSVP

As has been shown in Figure (5.3), when ECTP calls RSVP, the ECTP QoS parameters are mapped onto the RSVP TSpec traffic descriptors such as *p, r, b, m* and *M*. The TSpec parameter values are delivered from the sender to receivers via RSVP PATH message.

Upon arrival of the RSVP PATH message at a receiver, the information in the Sender_TSpec (TSpec) and ADSPEC objects will be passed across the RSVP API to the ECTP API and then to the application. The application interprets the arriving information, and uses it to select the resource reservation parameters. The parameters are composed into RSVP FLOWSPEC object (RSpec) and transmitted to the sender via the RSVP RESV messages. During the ECTP connection, each RSVP module reports a status on the network resource reservation to its associated ECTP daemon via RAPI. Such a status will initiate whether the reservation has been done successfully, or an error condition has been occurred. The ECTP protocol core can obtain these status codes by using RSVP *rapi_getfd()* and *rapi_dispatch()* from the RSVP daemon.

Once the ECTP sender makes the resource reservation, the negotiation operation will be continued to transmit packets to all receivers. The QoS monitoring and maintenance operations are the same as was explained in Chapter 4, section (4.4.2) and (4.4.3).

As has defined in Chapter 4, section (4.2.2), sender proposes *CHQ, OT* and *LQA* target values and the modified values by receivers must be set in the range of *CHQ* and *LQA*. Table (5.5) shows the target values for the corresponding ECTP QoS parameters.

ECTP QoS PARAMETERS	TARGET VALUES	DESCRIPTION
Throughput	LQA Throughput OT Throughput CHQ Throughput	LQA throughput ≤ OT Throughput ≤ CHQ Throughput
Transit delay	LQA Transit delay OT Transit delay	OT Transit delay ≤ LQA Transit delay
Transit delay jitter	LQA Transit delay jitter OT Transit delay jitter	OT Transit delay jitter ≤ LQA Transit delay jitter
Data loss	LQA Data loss OT Data loss	OT Data loss ≤ LQA Data loss

Table (5.5): ECTP QoS targets values for standard RSVP

5.4.2 MAPPING ARCHITECTURE

1. The sender proposes target values via CR packet to all active receivers.
2. Before the sender transmits CR packet, the ECTP daemon invokes its associated RSVP daemon through RAPI to map the ECTP QoS parameter target values into RSVP object traffic descriptors, and transmits these target values via an RSVP PATH message to the all receivers for negotiation.

ECTP QoS parameters target values such as throughput (*CHQ, OT* and *LQA*) and transits delay (*OT* and *LQA*), can be mapped into RSVP objects traffic descriptors to negotiate the resource reservation. Based on the QoS parameter values, RSVP objects can configure their traffic descriptors required for specifying the system and network resource if it is enabled in the network.

The mapping scheme assumes the non-negotiation mode in which the QoS parameter values proposed by sender are enforced to all receivers without negotiation.
However, the rest of ECTP QoS parameters such as delay jitter and data loss rate cannot be explicitly supported by RSVP, since RSVP signaling just provides a strict guarantee of transit delay by using the reservation of a bandwidth corresponding to the throughput.

The following are the mapping schemes for the throughput and transit delay parameters:

- **MAP "THROUGHPUT TARGET VALUES" INTO SENDER_TSPEC TRAFFIC DESCRIPTORS**

The mapping procedures are as follows:

- CHQ is directly mapped onto *p* in TSpec:
  CHQ throughput=p

- OT is directly mapped onto *r* in TSpec:
  OT throughput=r

- Token bucket (*b*) size in TSpec is set arbitrarily based on ECTP throughput parameters:
  CHQ throughput- OT throughput=b

- The reservation (*R*) in RSpec is sent to the mean value of ECTP throughput parameters:
  (CHQ throughput-OT throughput)/2=R

- The ECTP fixed header and ECTP MSS is directly mapped into *m* and *M* in TSpec respectively and sets as default.

After mapping, RSVP constructs the PATH message that contains TSpec parameters (*p, r, b, m* and *M*) based on mapped ECTP QoS parameter target values set by sender and ADSPEC. RSVP transmits the PATH messages to the receivers periodically. If an RSVP PathErr is indicated, the RSVP reports the corresponding statue codes to sender. When receivers receive the PATH message, which contains mapped target values from the sender, the RSVP daemon invokes its associated ECTP daemon.
Based on the PATH message received from the sender, the ECTP maps the following target values required by the receivers into the RSVP RSpec traffic descriptor.

- **MAP "THROUGHPUT TARGET VALUES" INTO FLOWSPEC TRAFFIC DESCRIPTOR**

$$\frac{(CHQthrough\ put - OTthroughp\ ut)}{2} = R$$

where *R* is the reserved bandwidth in RSpec.

- **MAP " TRANSIT DELAY TARGET VALUES" INTO FLOWSPEC TRAFFIC DESCRIPTOR**

  - OT is set to the required end-to-end delay (*R* in RSpec):
    OT transit delay= R

  - The difference between LQA and OT transit delay is set to slack term (*S*) in RSpec:
    LQA transit delay - OT transit delay= S

After mapping, RSVP constructs a RESV message that contains RSpec parameters based on the mapped ECTP QoS parameters target values set by receivers. Receivers transmit the corresponding RESV messages to the sender. When a RSVP RESV error is indicated, the receiver reports the corresponding status code to ECTP protocol.

Figure (5.6) and (5.7) illustrate the details of interfaces between ECTP and Standard RSVP during the connection lifetime.

SENDER

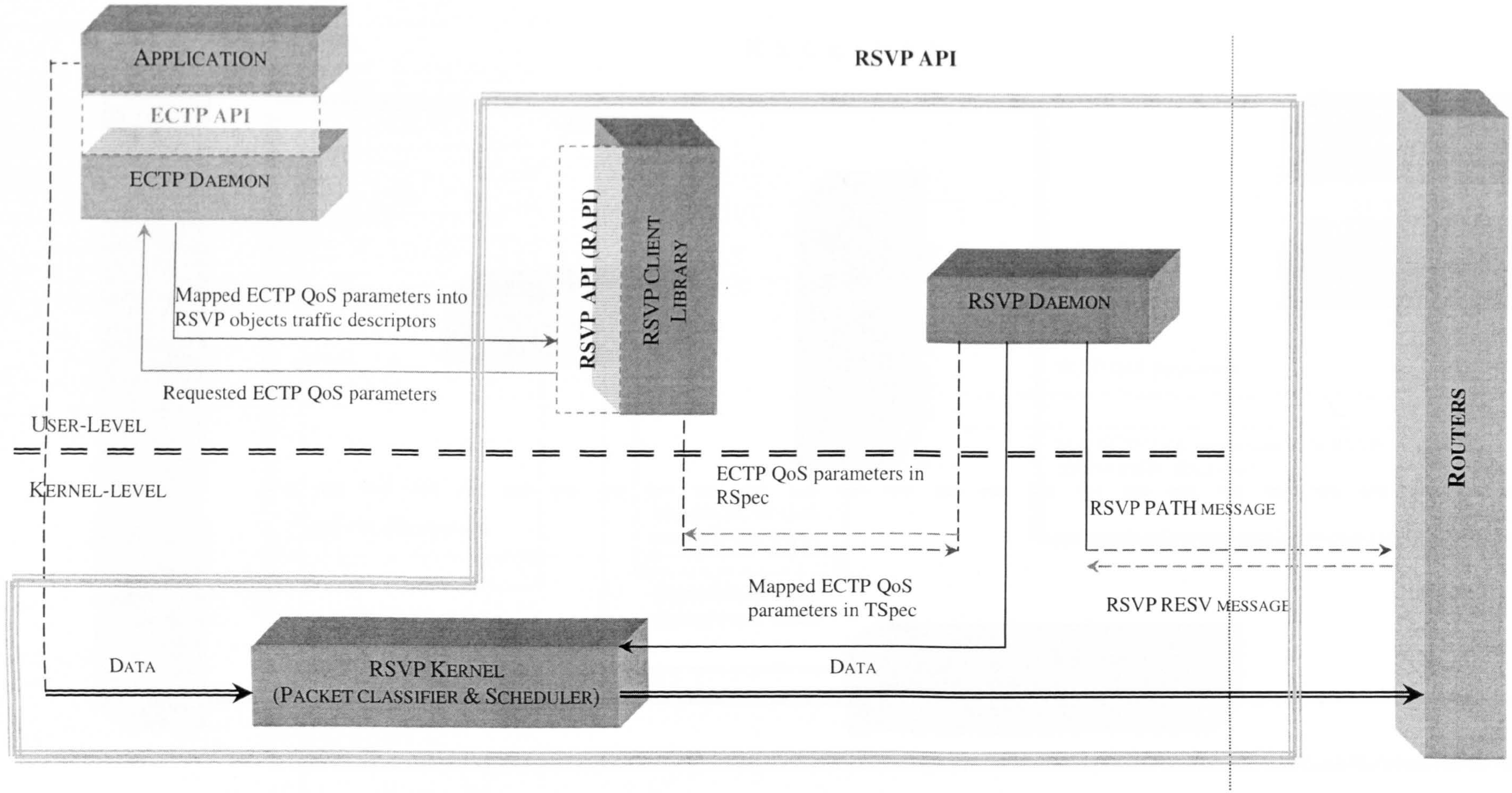


Figure (5.6)- Sender architecture of mapped ECTP QoS parameters into RSVP Objects

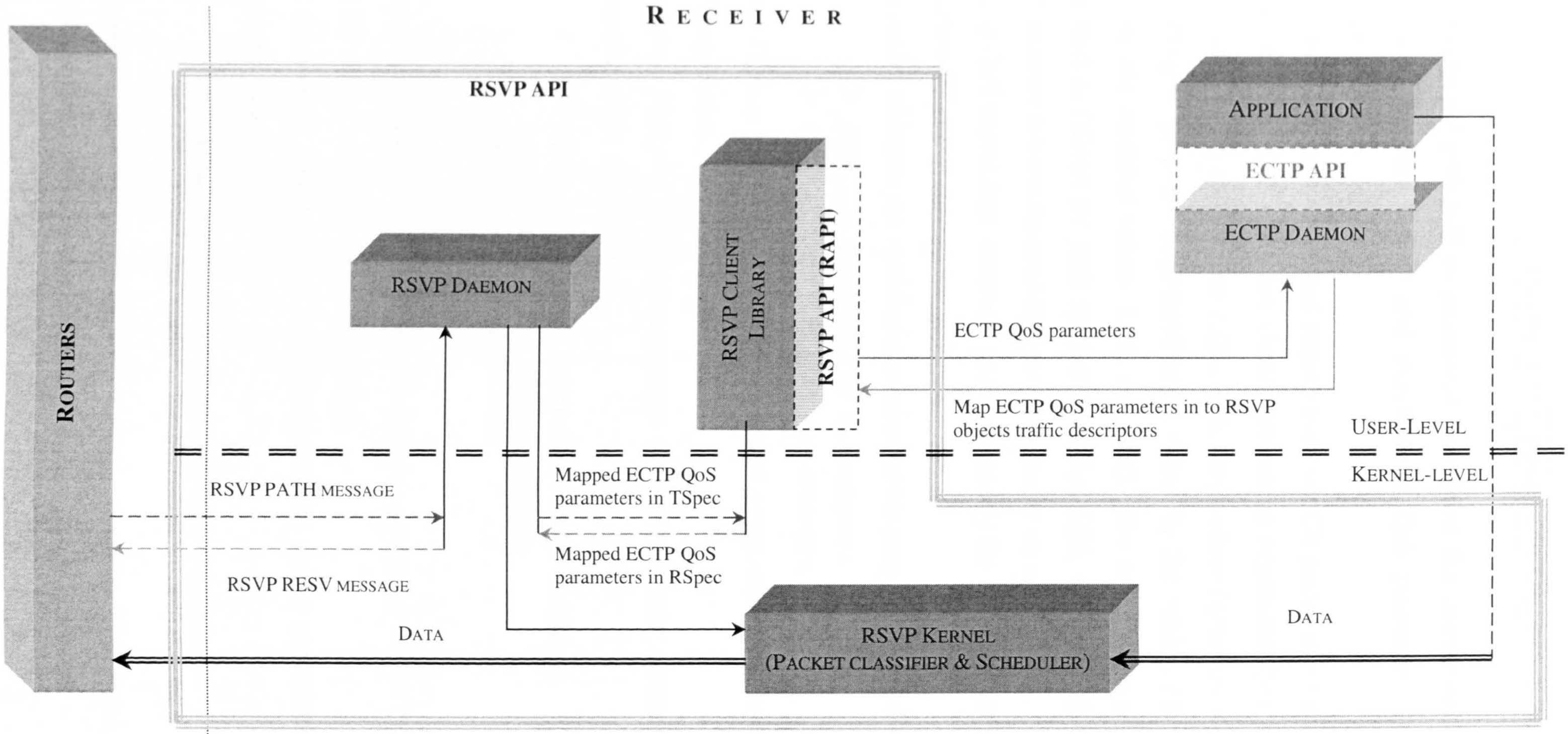


Figure (5.7)-Receiver architecture of mapped ECTP QoS parameters into RSVP Objects

3. The end-to-end path will be guaranteed throughout the connection, after negotiated agreements between sender and receivers on QoS parameters that required by application.
4. The sender proposes negotiated target values via CR packet to all receivers. The CR packet contains QoS extension element indicating negotiated QoS parameter target values required by applications: LQA_0, OT_0, CHQ_0 where $LQA_0<OT_0<CHQ_0$.

   According to proposed negotiated target values by the sender, each receiver i can propose the modified values: LQA_i and CHQ_i. The modified target values can be described as follows for each receiver i: $LQA_0< LQA_i <OT_0<CHQ_0< CHQ_i$. Finally each receiver delivers the modified values to sender via a CC packet.

   During QoS negotiation, receivers must not modify the OT value for each parameter, therefore OT_0 value stay unchanged.
5. The sender arbitrates the modified parameter values proposed by receivers as follows:

   - LQA_{max} = max (LQA_i) for each receiver i
   - CHQ_{min} = min (CHQ_i) for each receiver i,

   where LQA_{max} and CHQ_{min} are the negotiated target values that have resulted from QoS negotiation.

   For delay, jitter and data loss rate target values, the sender arbitrates the modified parameter values as follows:

   - LQA_{min} value would be obtained instead of LQA_{max}
     LQA_{min}= min (LQA_i), for each receiver i
   - CHQ value won't be not used
   - OT value<LQA value

6. The sender announces the negotiated throughput parameter values: LQA_{max}, CHQ_{min}, and OT values to receivers via HB packets.

   For delay, jitter and data loss rate parameters, the sender announces the negotiated parameter values as follows:

   - LQA_{min} value will be obtained instead of LQA_{max}
   - OT value<LQA value

7. QoS negotiation is not performed for late-joining receivers. The sender just notifies the negotiated target values to the late-joiner receiver via a JC packet.

5.5 MODEL (II)- ENABLING ECTP PROTOCOL TO NEGOTIATE END-TO-END RESOURCE RESERVATION USING THE STANDARD AND AGGREGATED RSVP SIGNALING PROTOCOL.

5.5.1 DESCRIPTION

The problem with the design of standard RSVP [BI01] is in its applicability statement, i.e. "*it lacks facilities for aggregation of individual reserved sessions into a common class therefore use of such aggregation is required for scalability*" [Bernet00]. In the standard RSVP, per-flow reservation, may require a non-trivial amount of message exchange, computation, and memory resource in each router along the way. It would be better to reduce this to a more manageable level where load is heaviest and aggregation is possible. This model proposes to map ECTP QoS parameters target values into the standard and aggregated RSVP object traffic descriptors. Figure (5.8) illustrates interworking between ECTP with the standard and aggregated RSVP. [CS92]

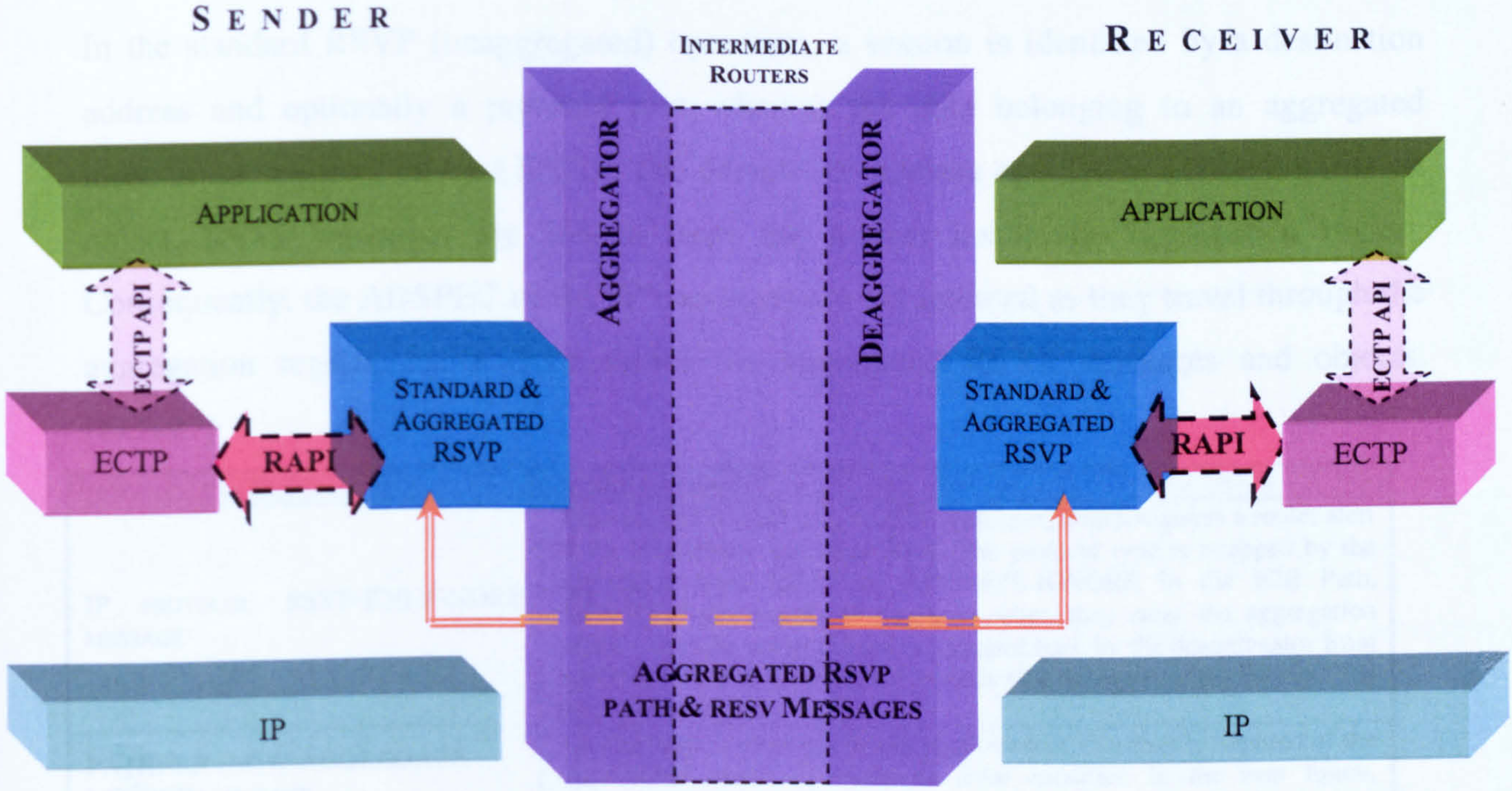


Figure (5.8): Interworking of ECTP with standard and aggregated RSVP

Aggregation depends on being able to hide E2E RSVP messages from RSVP-capable routers inside the aggregation region. To achieve this end, the IP protocol number in the

E2E reservation's PATH, PATHTear, and ResvConf messages is changed from standard RSVP to RSVP-E2E-IGNORE upon entering the aggregation region, and restored to RSVP at the deaggregator point.

These messages are ignored (no state is stored and the message is forwarded as a normal IP datagram) by each router within the aggregation region whenever they are forwarded to an interior interface. As it discussed in Chapter (3), section (3.9.2), the RSVP objects (Sender_TSpec and FLOWSPEC) of E2E reservations are summed into the corresponding information elements in aggregate PATH and RESV messages, which are as follow [MB+97]:

- Aggregate PATH messages are sent from the aggregator to the deaggregator (s) using RSVP's normal IP protocol number. The change of the protocol number disables reservations inside the aggregation region.

- Aggregate RESV messages are sent back from the deaggregator to the aggregator, thus establishing an aggregate reservation on behalf of the set of E2E flows that use this aggregator and deaggregator.

In the standard RSVP (unaggregated) operation, a session is identified by a destination address and optionally a protocol port whereas the data belonging to an aggregated reservation is identified by a DSCP. The destination address and DSCP define the session object. RSVP messages are hidden from the routers inside the aggregation region. Consequently, the ADSPEC of RSVP messages are not updated as they travel through the aggregation region. Table (5.8) shows the aggregated RSVP messages and objects. [BO+97]

AGGREGATED RSVP	DESCRIPTIONS
IP PROTOCOL RSVP-E2E-IGNORE MESSAGE	This element is used only on E2E messages, which requires a router alert (Path, PATHTear and ResvConf). The protocol type is swapped by the aggregator from RSVP to RSVP-E2E-IGNORE in the E2E Path, PATHTear and ResvConf messages when they enter the aggregation region. Also the protocol type is swapped back by the deaggregator from RSVP-E2E-IGNORE to RSVP in such E2E messages when they exit the aggregation region.
PATHERR (NEW-AGGREGATE-NEEDED) MESSAGE	This element is required, which signifies that an action is required of the aggregation router to avoid an error condition in the near future. However this does not signify that a fatal error has occurred.
SESSION object	It contains of two values: the IP address of the aggregate session destination, and the DSCP that it will use on the E2E data the reservation contains.
SENDER_TEMPLATE object	It identifies the aggregating router for the aggregate reservation.
FILTER_SPEC object	It identifies the aggregating router for the aggregate reservation, and it syntactically identical to the Sender_Template.

Table (5.8): Descriptions of aggregated RSVP messages and objects

As was proposed and discussed in section (5.4), ECTP QoS parameters can be mapped into standard RSVP object traffic descriptors. In this section, we consider the mapping of ECTP QoS parameters into the aggregated RSVP objects' traffic descriptors. This model has a similar architecture to the model (I). However, the aggregated RSVP may use new messages and objects (Table 5.8). As mentioned in section (3.5), ECTP specification has defined three target values to map into the standard RSVP objects: *CHQ, LQA* and *OT*. [BV98]

In aggregated RSVP, it has been assumed that no aggregated path exists between the aggregator and deaggregator. As a result, model (II*)* proposes a new target value: Mean quality (MQ) target value. It is described differently for each QoS parameters:

- MQ for throughput is the mean value of CHQ and LQA

$$MQ\ throughput = \frac{CHQ + LQA}{2}$$

- MQ for transit delay, jitter and data loss is the mean value of minimum and maximum of LQA.

$$MQ\ transit\ delay = \frac{\max(LQA) + \min(LQA)}{2}$$

The sender proposes *OT, MQ* and *LQA* target values and the modified values by receivers must be set in the range of *MQ*. Table (5.9) shows the target values for the corresponding ECTP QoS parameters.

ECTP QoS PARAMETERS	TARGET VALUES	DESCRIPTION
Throughput	LQA Throughput OT Throughput MQ Throughput	LQA throughput ≤ OT Throughput ≤ MQ Throughput
Transit delay	MQ Transit delay OT Transit delay	OT Transit delay ≤ MQ Transit delay
Transit delay jitter	MQ Transit delay jitter OT Transit delay jitter	OT Transit delay jitter ≤ MQ Transit delay jitter
Data loss	MQ Data loss OT Data loss	OT Data loss ≤ MQ Data loss

Table (5.9): ECTP QoS targets values for aggregated RSVP

5.5.2 MAPPING ARCHITECTURE

1. The Sender proposes target values via CR packet to all receivers.
2. Before transmitting the CR packet, the ECTP daemon invokes its associated RSVP daemon through RAPI to map the ECTP QoS parameter target values into RSVP object traffic descriptors.

   ECTP QoS parameters target values such as throughput (*LQA*, *MQ* and *OT*) and transit delay (*OT* and *MQ*), can be mapped into RSVP objects traffic descriptors to negotiate the resource reservation. Based on the QoS parameter values, RSVP objects can configure their traffic descriptors required for specifying the system and network resource if it is enabled in the network.

   The mapping scheme assumes the non-negotiation mode in which the QoS parameter values proposed by sender are enforced to all receivers without negotiation.

   However, the rest of ECTP QoS parameters such as delay jitter and data loss rate cannot be explicitly supported by RSVP, since RSVP signaling just provides a strict guarantee of transit delay by using the reservation of a bandwidth corresponding to the throughput. The following are the mapping schemes for the throughput and transit delay parameters:

   - **MAP "THROUGHPUT TARGET VALUES" INTO SENDER_TSPEC TRAFFIC DESCRIPTORS**

     The mapping procedures are as follows:

     - *MQ throughput = p*

       where *p* is the peak rate in TSpec,
     - *OT throughput = r*

       where *r* is the token rate in TSpec,
     - *(MQ throughput-OT throughput)= b*

       where *b* is the token bucket size, which is set arbitrarily, based on ECTP throughput parameters.
     - The ECTP fixed header and ECTP MSS is directly mapped into *m* and *M* in *TSpec* respectively and sets as default.

After mapping, RSVP constructs the PATH message that contains TSpec parameters (*p*, *r*, *b*, *m* and *M*) based on mapped ECTP QoS parameter target values

set by sender and ADSPEC. Before the aggregator forwards the Path message, it modifies its IP protocol Number to RSVP-E2E-IGNORE region or IP tunnel.
Path message forwarded by aggregator with TSpec object contains ECTP QoS parameters. If no aggregated Path exists, the deaggregator sends a New-Agg-needed PathErr to aggregator to set up aggregate Path states for supported DSCP(s). The aggregator follows the request from the deaggregator and signals an aggregator Path (New-Agg-needed PathErr code) for DSCP.

When receivers receive the PATH message, which contain the mapped target values from the sender, the RSVP daemon invokes its associated ECTP daemon. Based on the PATH message received from the sender, the ECTP mapped the following target values required by the receivers into the RSVP RSpec traffic descriptor:

- **MAP "THROUGHPUT TARGET VALUES" INTO FLOWSPEC TRAFFIC DESCRIPTOR**

$$\frac{(MQthroughput - OTthroughput)}{2} = R$$

where R is the reserved bandwidth in RSpec.

- **MAP " TRANSIT DELAY TARGET VALUES" INTO FLOWSPEC TRAFFIC DESCRIPTOR**

  - OT is set to the required end-to-end delay (R in RSpec)
    OT transit delay= R

  - The difference between MQ and OT transit delay is set to slack term (S) in RSpec
    MQ transit delay - OT transit delay= S

After mapping, RSVP constructs RESV message that contains RSpec parameters based on the mapped ECTP QoS parameters target values set by receivers. Receivers transmit the corresponding RESV messages to the deaggregator. When a RSVP RESV error is indicated, the receiver reports the corresponding status code to ECTP protocol.

The deaggregator immediately proceeded with establishment of aggregate reservations for DSCP(s). The deaggregator includes the optional ResvConfirm request in these aggregate RESV. The aggregator merely complies with the received ResvConfirm request and returns the corresponding aggregate ResvConfirm.

The deaggregator has explicit (clear) confirmation that both aggregate RESV are established. The deaggregator applies the mapping policy defined by the network administrator to map E2E RESV onto an aggregate reservation. The deaggregator performs admission control of the E2E Resv onto the aggregate RESV for DSCP.

When the aggregate RESV for DSCP has been established with sufficient bandwidth to support the E2E RESV. The deaggregator forwards the E2E RESV to the aggregator including a DCLASS object conveying the selected mapping onto DSCP. The aggregator records the mapping of E2E RESV onto DSCP. The aggregator removes the DCLASS object and forwards the E2E Resv towards the sender.

Figure (5.8), (5.9) and (5.10) illustrate the details of interfaces between ECTP and the standard and aggregated RSVP during the connection lifetime.

SENDER

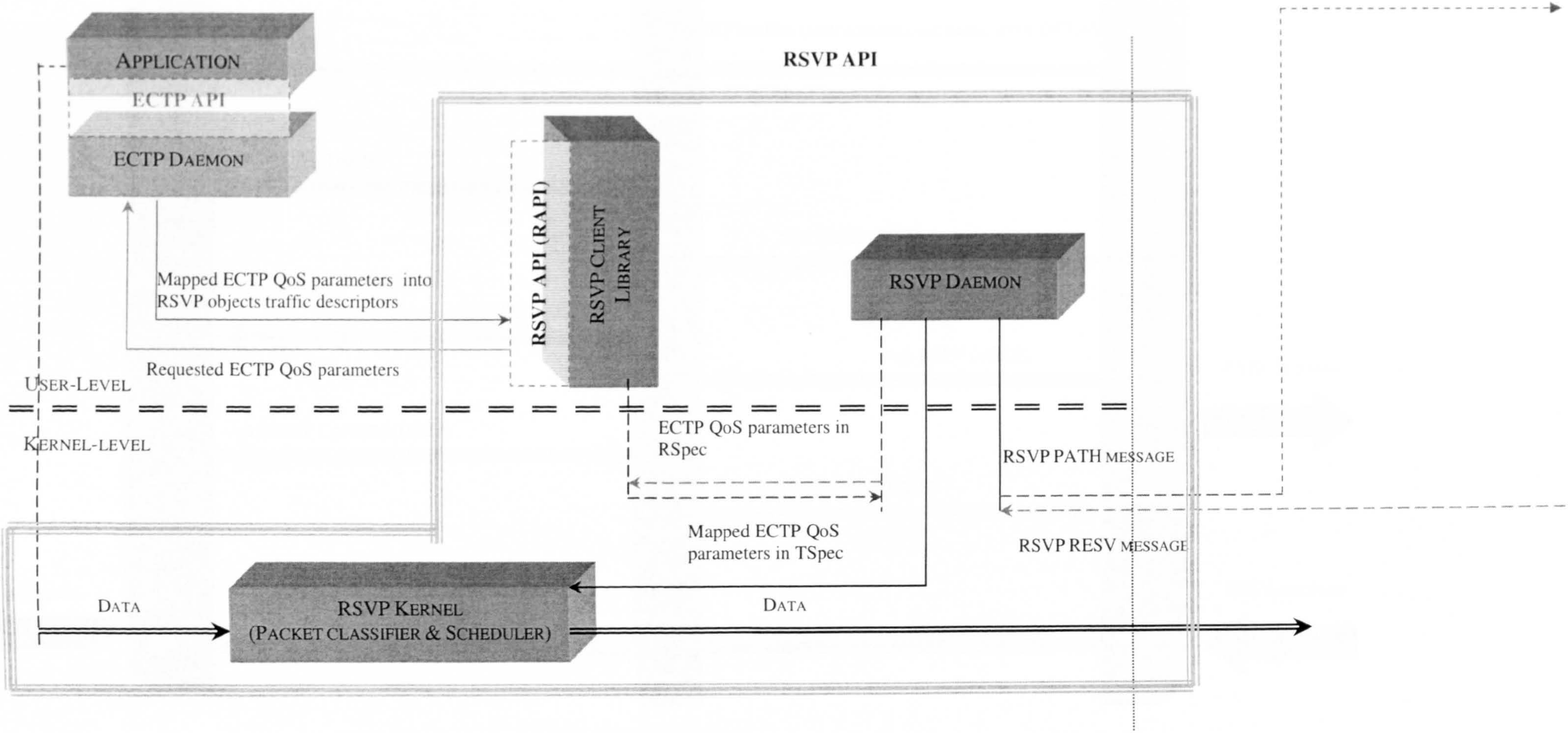


Figure (5.9)- Sender architecture of mapped ECTP QoS parameters into RSVP Objects

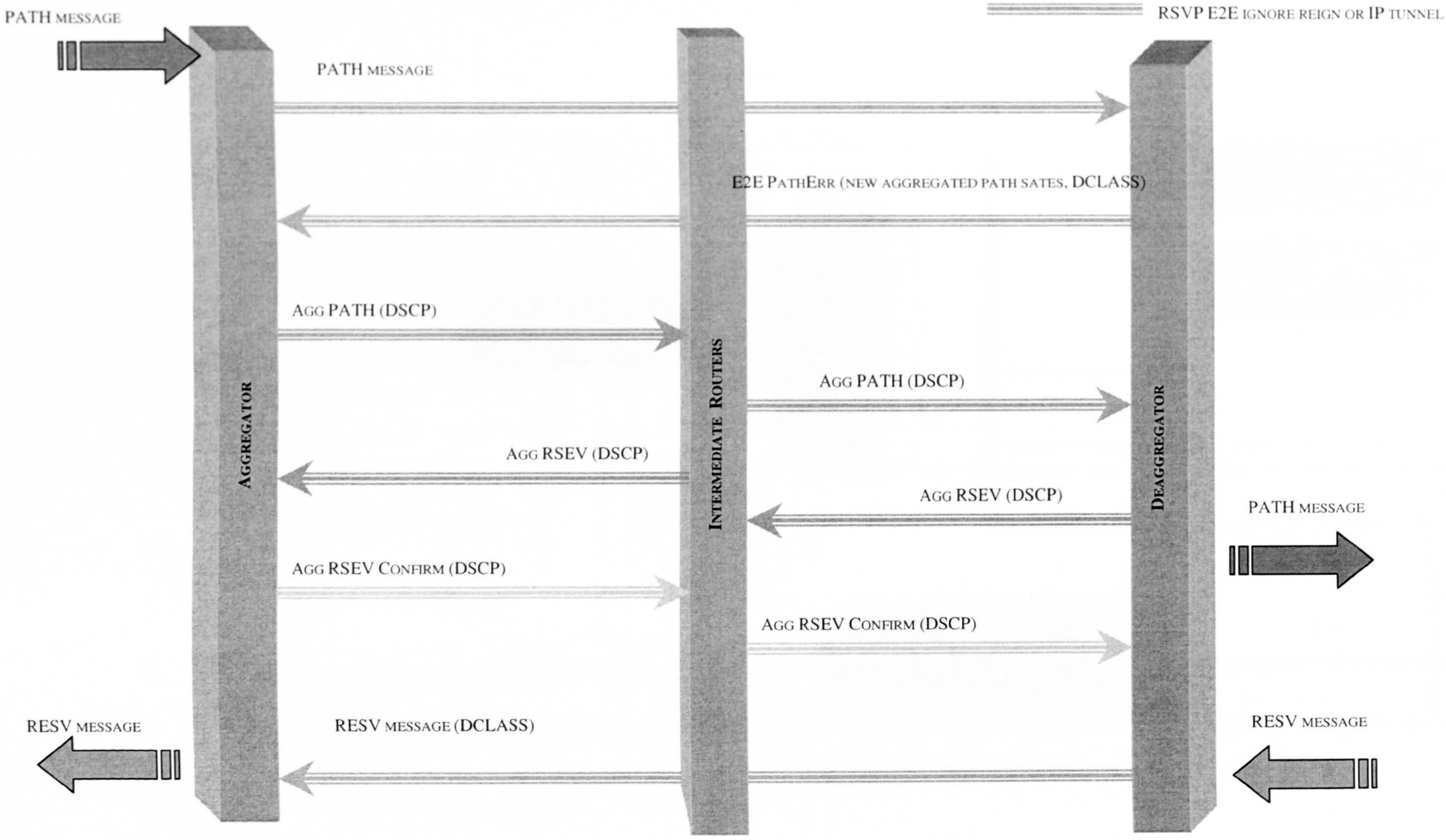


Figure (5.10): Aggregator and Deaggregator routers

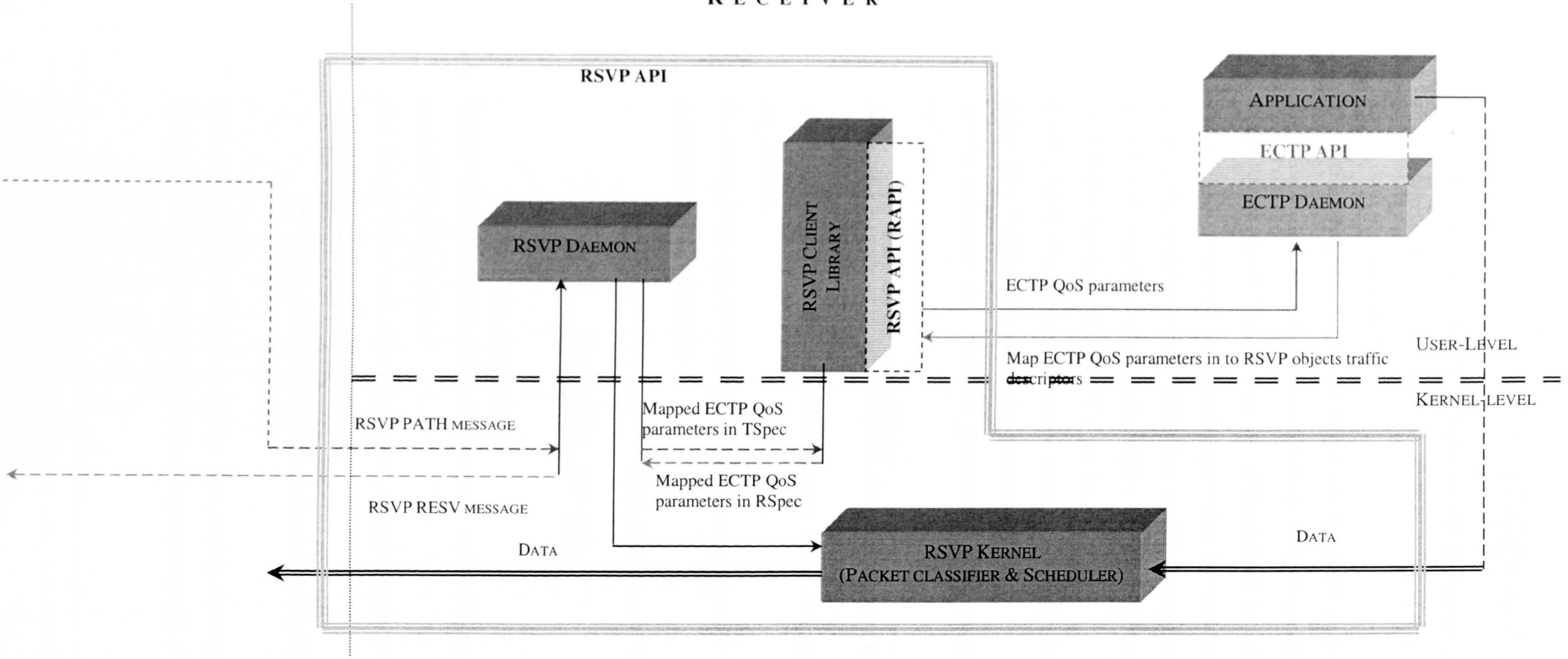


Figure (5.11)- Receiver architecture of mapped ECTP QoS parameters into RSVP Objects

3. After negotiated agreements between sender and receivers on QoS parameters that are required by applications, the end-to-end path will be guaranteed throughout the connection,
4. The sender proposes negotiated target values via CR packet to all receivers. The CR packet contains QoS extension element indicating negotiated QoS parameter target values required by applications: MQ_0 and OT_0 where $OT_0<MQ_0$.

   According to proposed negotiated target values by the sender, each receiver *i* can propose the modified values: MQ_i. The modified target values can be described as follows for each receiver *i*: $OT_0<MQ_0< MQ_i$. Finally each receiver delivers the modified values to sender via a CC packet.

   During QoS negotiation, receivers must not modify the *OT* value for each parameter, therefore the OT_0 value stay unchanged.
5. Sender arbitrates the modified parameter values proposed by receivers as follows:

$$MQ_{min/max} = \frac{\max(LQA_i) + \min(CHQ_i)}{2} \text{ for each receiver } i,$$

   where $MQ_{min/max}$ is the negotiated target value that has resulted from QoS negotiation.

   For delay, jitter and data loss rate target values, the sender arbitrates the modified parameter values as follows:

   - MQ_{min} value would be obtained instead of MQ_{max}

$$MQ_{min} = \frac{\max(LQA_i) + \min(LQA_i)}{2}$$

   - *OT* value<*MQ* value

6. Sender announces the negotiated parameter value: $MQ_{min/max}$ and *OT* values to receivers via HB packets.

   For delay, jitter and data loss rate parameters, the sender announces the negotiated parameter values as follows:

   - MQ_{min} value would be obtained instead of MQ_{max}
   - *OT* value<*MQ* value

7. QoS negotiation is not performed for late-joining receivers. Sender just notifies the negotiated target values to the late-joiner receiver via JC packet.

5.6 SUMMARY

This chapter has proposed two models to map ECTP QoS parameters into standard and aggregated RSVP signaling protocol

- **MODEL (I)-** Enabling ECTP transport protocol to negotiate end-to-end resource reservation using the standard RSVP signaling protocol (IntServ).

- **MODEL (II)-** Enabling ECTP transport protocol to negotiate end-to-end resource reservation using the standard and aggregated RSVP signaling protocol (IntServ and DiffServ).

The model (I) has proposed to map ECTP QoS parameters into the standard RSVP signaling protocol to enable ECTP transport protocol with QoS management. This model uses RSVP and ECTP APIs to interact with each other and with the applications respectively. However because of lack of scalability in standard RSVP throughout the core network, the model (II) has been proposed which uses aggregated RSVP.

This model has proposed to map ECTP QoS parameters into the aggregated RSVP signaling protocol to enable ECTP transport protocol with the end-to-end QoS management. Therefore this model is enabled with a dynamic admission control and scalability throughout the core network.

The next chapter proposes and defines the process models to implement the ECTP transport protocol and the two proposed models (model (I) and (II)) in the OPNET (Optimized Network Engineering Tool) simulation tool. The implementations follow the OPNET modeling methodology to define each process model.

CHAPTER 6

DESIGN AND IMPLEMENTATION

6. OVERVIEW

As it mentioned, Chapter 5 has proposed and discussed two models (model (I) and (II)) to enable the ECTP transport protocol to operate with the standard and/or aggregated RSVP resource reservation. The proposed models enabled the ECTP transport protocol to provide end-to-end QoS for multicast applications. This chapter implements ECTP transport protocol and the two proposed models in the OPNET simulation tool (8.1). The Optimised Network Engineering Tool (OPNET) simulator provides a comprehensive development environment for modelling and performance of communications networks. (Appendix C)

6.1 OPNET ARCHITECTURE

The OPNET package consists of number of tools, each one focusing on particular aspects of modelling task. These tools fall into three major categories that correspond to the three phases of modelling and simulation projects: *Specification, Data Collection and Simulation* and *Analysis*. Figure (6.1) shows the simulation project cycle in OPNET simulation tool. [OPNET_Concept]

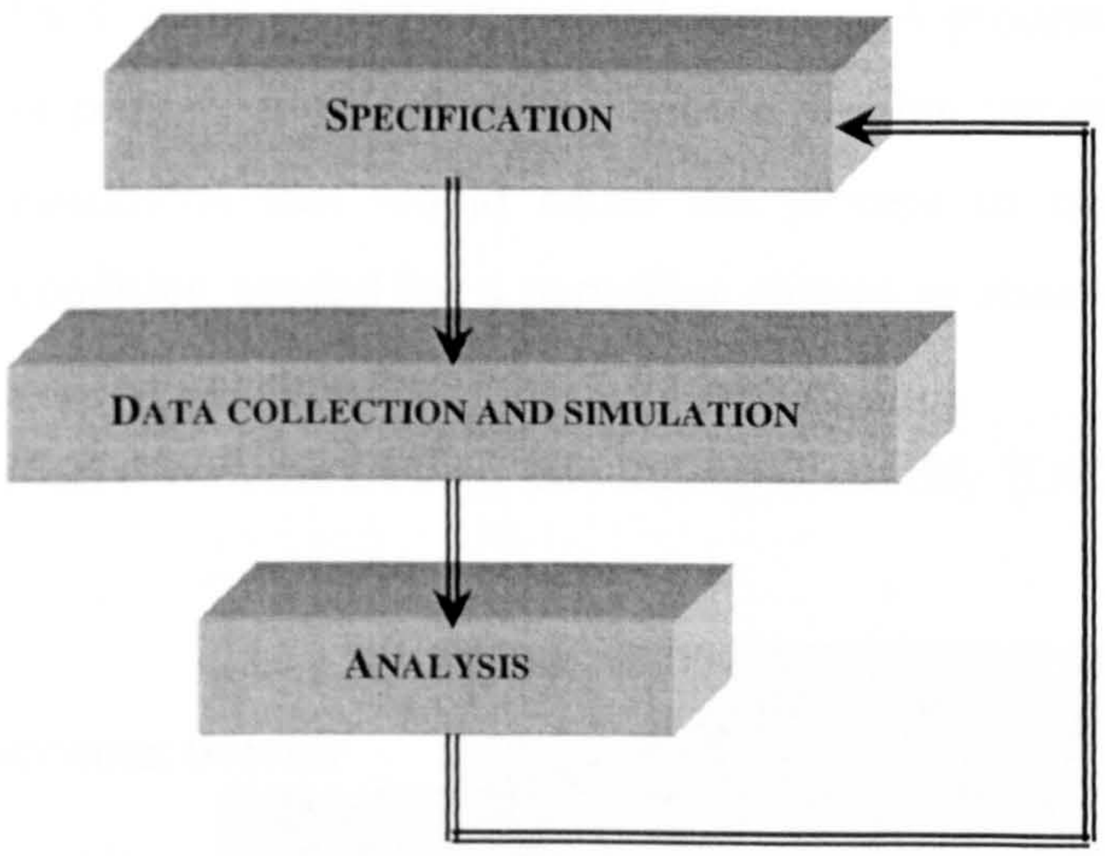


Figure (6.1): Simulation Project Cycle

6.1.1MODELLING DOMAINS

To provide useful data, network models must combine accurate descriptions of topology, data flow, and control flow. Since no single paradigm of visual

representation is ideally suited for all three of these model types, OPNET utilises separate model formats for each. To lend structure and discipline to the overall model, OPNET models fit together in a hierarchical fashion. The following are the OPNET modelling hierarchy [OPNET_Concept]:

- **NETWORK MODEL,** which defines the position and interconnection of communicating entities, or nodes. Each node is described by a block structured data flow diagram.
- **NODE MODEL,** which typically depicts the interrelation of processes, protocols, and subsystems. Each programmable block in a Node Model has its functionality defined by process model.
- **PROCESS MODEL,** which combines the graphical power of a state-transition diagram with the flexibility of a standard programming language and a broad library of pre-defined modelling functions. OPNETs process editor expresses process models in a language called Proto-C, which is specifically designed to support development of protocols and algorithms. Proto-C is based on a combination of state transition diagrams (STDs), a library of high-level commands known as kernel Procedures (KPs), and the general facilities of the C and C++ programming Language. A process model's STD defines a set of primary modes or states that the process can enter and, for each state, the conditions that would cause the process to move to another state. The condition needed for a particular change in state to occur and the associated destination state are called a transition.

Figure (6.2) shows the OPNET Modelling Hierarchy. [OPNET_Methodology],

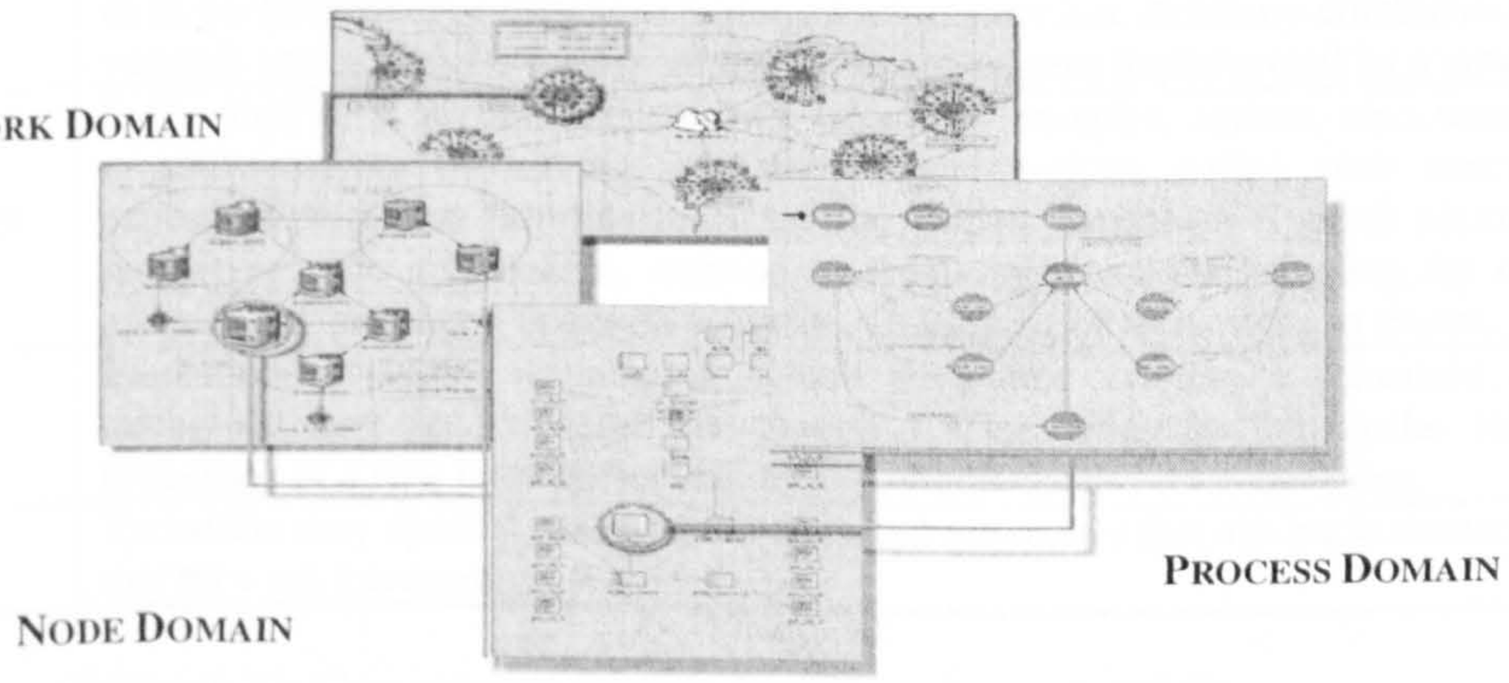


Figure (6.2): OPNET Modelling Hierarchy

6.1.2 PROCESS MODEL ARCHITECTURE

As mentioned earlier, in the node domain the modules are user-programmable elements that are key elements of communication nodes. The tasks that these modules execute are called processes. Processes in OPNET are based on process models that are defined in the Process Editor. Just as nodes created in the Project Editor are instances of node models defined with the Node Editor, each process that executes in process modules.

The process-modelling paradigm of OPNET supports the concepts of process groups, which consist of multiple processes that execute within the same modules. When a simulation begins, each module has only one root or parent process, which creates another process, is called child process. Process models are expressed in Proto-C, a language combining graphical state-transition diagrams (STDs), Embedded C/C++ language data items and statements, and a library of Kernel Procedures (KPs) that provide commonly needed functionality for modelling communications and information processing systems. [OPNET_ Process]

6.1.3 STATE TRANSITION DIAGRAM (STD)

The state transition diagram in Proto-C decomposes the states of the system and the processing that should take place at each interrupt. Table (6.1) describes components of STD. [OPNET_STD]

STD COMPONENT	DESCRIPTION
STATE VARIABLES	Processes maintain private state variables with named variables of arbitrary data types including OPNET-specific, general C/C++ language, and user-defined types. This capability allows a process to flexibly maintain counters, routing tables, statistics related to its performance, or messages requiring retransmission. Arbitrary combinations of state variable values may be used in all decisions and actions implemented by a process.
STATE EXECUTIVES	Each state of a process can specify arbitrarily complex actions associated with the process entering or leaving that state. These actions, called *state executives*, are expressed with the full flexibility of the C/C++ language. Typical actions include modifying state information, creating or receiving messages, updating the contents of and sending messages, updating statistics, and setting or responding to timers.
TRANSITIONS CONDITIONS	Transitions condition statements, which determine whether a transition should be traversed, may be expressed as general C/C++ language that make reference to properties of a new interrupt as well as to combinations of state variables.
TRANSITION EXECUTIVE	Transition may specify general actions, called *executives* that are implemented each time that they are traversed.

Table (6.1): Components of state transition diagram (STD)

6.1.4 KERNEL PROCEDURES (KPS)

The simulation kernel services can be accessed through Kernel Procedures (KPs), which are procedures that can be called from within process models. KPs are the OPNET-provided functions, which along with state transition diagrams (STDs) and Proto-C, supports the development of protocols and algorithms. All the KPs begin with prefix "op_".[OPNET_Kernel]

6.1.4.1 KPS VARIABLES AND DATA TYPES

In KPs, all the variables, regardless of their category, have a name, a datatype and a value. The name and datatype of a variable remain fixed throughout the life of a process. The value component refers to the actual contents of memory that is associated with a variable. For simple variables this may be an individual number or string of characters; for compound data structures many such items could be involved. Table (6.2) shows the essential OPNET KPs predefined data structures used in process model. [OPNET_ Process]

DATA STRUCTURE	DESCRIPTION	DATA TYPE
Compare codes	indicate whether an operation completed successfully	COMPCODE
Distributions	describes the mapping of a random number into specific numeric outcome, consistent with a probability density function (PDF)	DISTRIBUTION
Event Handles	identifies a pending simulation event (interrupt)	EVHANDLE
Statistic handles	identity the global and local statistic that are created dynamically	STATHANDLE
Interface Control Information	are collections of structured data that used to associate additional user defined information with an event.	ICI
Lists	are ordered collection of data elements. Elements in the lists can range from simple C/C++ data types to complex data structures.	LIST
Object ID	is a numeric serial number, which uniquely identifies a single simulation object.	OBJID
Packets	structured group of data fields representing a message that can flow over packet streams (node domain) and over links (network domain).	PACKET
Log handles	when creating simulation logs to aid in simulation debugging or results analysis	LOG_HANDLE
Process Handles	identify an active process within a simulation	PROHANDLE

Table (6.2): OPNET Kernel procedures

Table (6.3) shows Proto-C processes in three distinct categories.

CATEGORIES	DESCRIPTION
State variables (SV)	Values defined as static variables in a process are visible to all states. It is equivalent to global variables in C programming
Temporary variables (TV)	Variables defined in this block are only visible within a state. In other words, the values in temporary variables set in a state are no longer meaningful once the process exits from that state.
Header block (HB)	This is similar to a header file in C programming. Related includes files and macros are defined in the header block, as well as the state transition conditions (in OPNET STD conditions are defined as macros)
Function Block (FB)	This block stores common produces. These procedures are help functions invoked by programs in enter-exec or exit-exec.
Diagnostic Block (DB)	OPNET simulations automatically incorporate a sophisticated "debugger" called OPNET Diagnostic Block (ODB) that allows the user to interactively monitor the progress of a simulation and the state of its object
Termination block (TB)	Producers in this block are invoked when the simulation terminates.

Table (6.3): Proto-C categories

6.2 IMPLEMENTATION OF ECTP SPECIFICATIONS ON OPNET

6.2.1 ECTP PROCESS MODEL

The proposed model follows the model development methodology in OPNET from the previous section. It assumes that the transport connection has been defined over the homogeneous environment, which means a condition in transmit diversity where all TS-users have agreed to a common set of transmit QoS values and so all sending TS-user transmit data at the same rate. Also it assumes the receivers have completed their enrolment process before connection creation (section 2.7).

6.2.2 ECTP MODEL ARCHITECTURE

Figure (6.3) illustrates the relevant portions of a typical node model where the ECTP model resides:

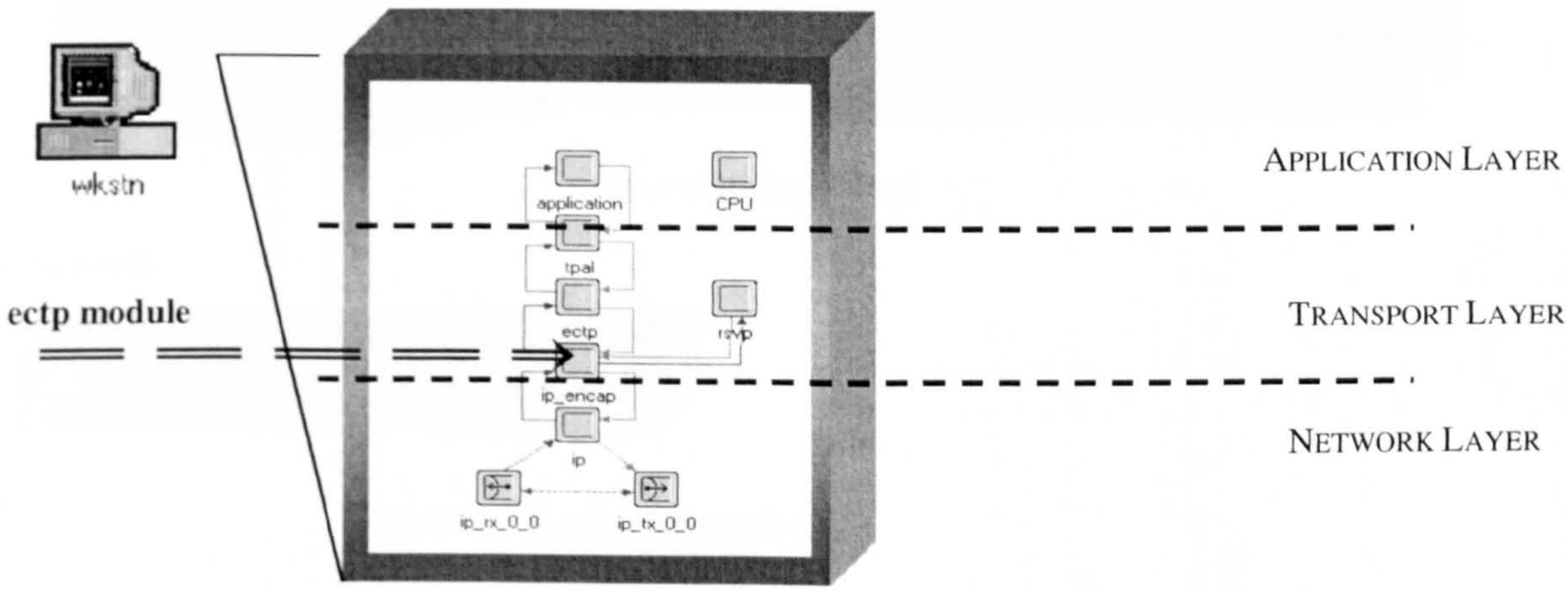


Figure (6.3): ECTP module in the Node model

Each Node intended to use ECTP has a "ectp" module. Table (6.4) summarizes the four ECTP process models.

ECTP PROCESS MODELS		
ectp_manager	Represents the parent (root) process of the "ectp" module. It manages a set of ECTP connections by invoking the appropriate ectp_connection to handle the connection and data transmission requests.	Parent process model
ectp_connection	Represents the ECTP connection process, which manages the connection phases. It is generated by ectp_manager process to manage activities related to an individual connection.	Child process model
ectp_QoS_management	Represents the ECTP QoS management process, which manages QoS level through out the connection.	Child process model
rsvp	Represents RSVP protocol process model, which reserves resources in ectp_QoS_management.	Child process model of ectp_QoS_managemen
aggrsvp	Represents aggregated RSVP protocol process model, which reserves resources in ectp_QoS_management.	Child process model of ectp_QoS_managemen

Table (6.4): ECTP process models

6.2.3 "ectp_manager "STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

Figure (6.4) illustrates a state transition diagram (STD) for ectp_ manager and defines the interdependent modules and communication transitions. The modules of ECTP connection are as follows:

- Connection creation process;
- Data transmission process;
- Connection termination process.

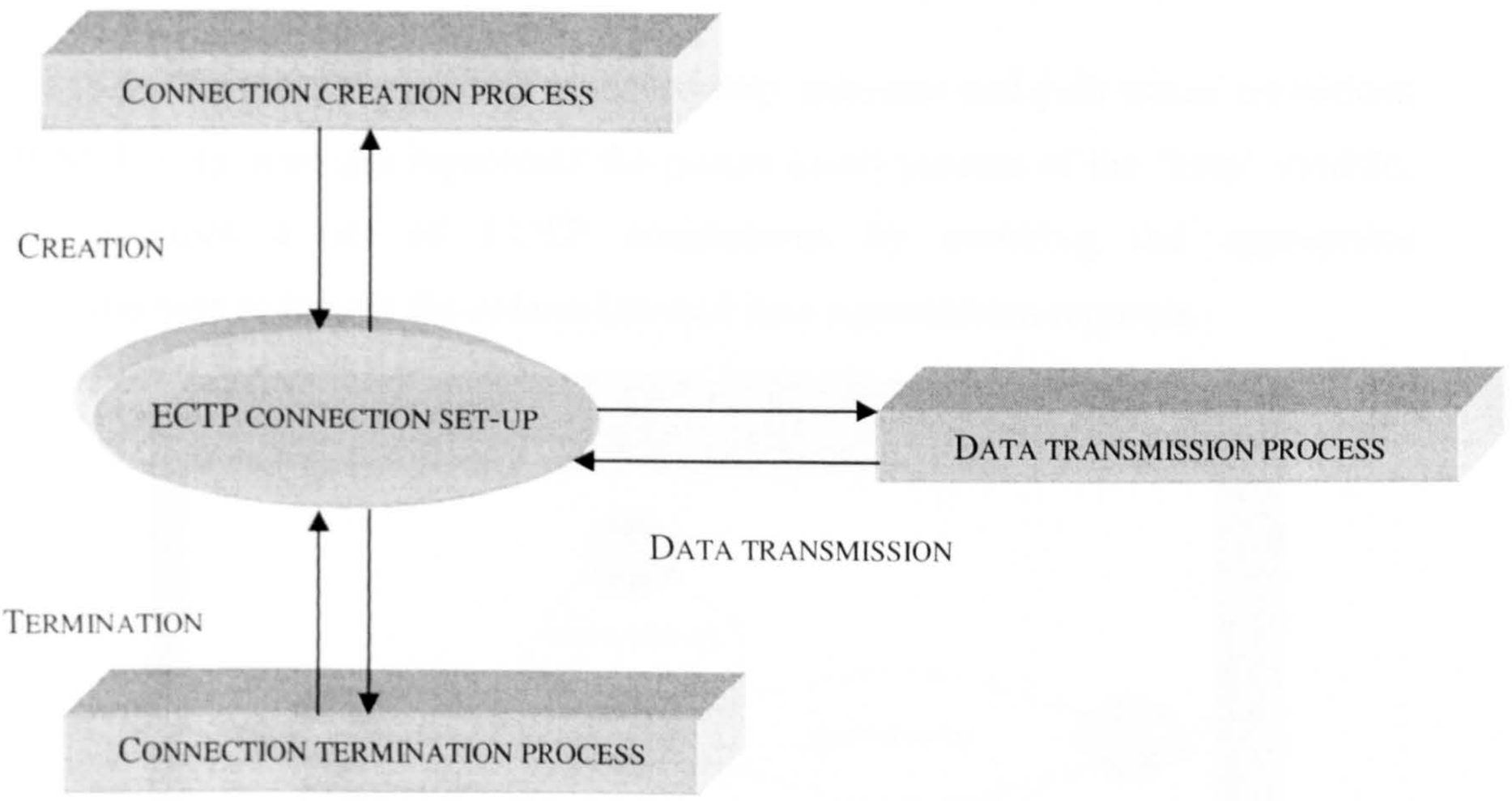


Figure (6.4): ectp_manager STD

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.5) describes the description of each state in each process.

STATE	DESCRIPTION
ECTP connection set up	All the members are enrolled into the multicast group and the connection session is created.
Connection Creation process	Sending a connection creation message to some or all the enrolled receivers whom may respond with confirmation messages to the sender. The connection creation is completed when the sender receives the confirmation messages from all of the active receivers
Data transmission process	After the connection is created, the sender begins to transmit multicast data.
Connection creation process	The termination messages are sent to all the receivers, after all the multicast data are transmitted.

Table (6.5): Description of each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.6) shows the Event response for all processes

STATE	LOCAL EVENTS	NEXT STATE
Init	Enrolled to the connection	Set-up
Set-up	Connection Creation Data Transmission Connection Termination	Create Transmission Termination
Create	Connection Creation	Set-up
Transmission	Data transmission	Set-up
Termination	Connection termination	Set-up

Table (6.6): Event response

Figure (6.5) illustrates the process model of ectp_manager and their states' transitions in OPNET. ectp_manager represents the parent (root) process of the "ectp" module, which manages a set of ECTP connections by invoking the appropriate ectp_connection to handle the connection and data transmission requests.

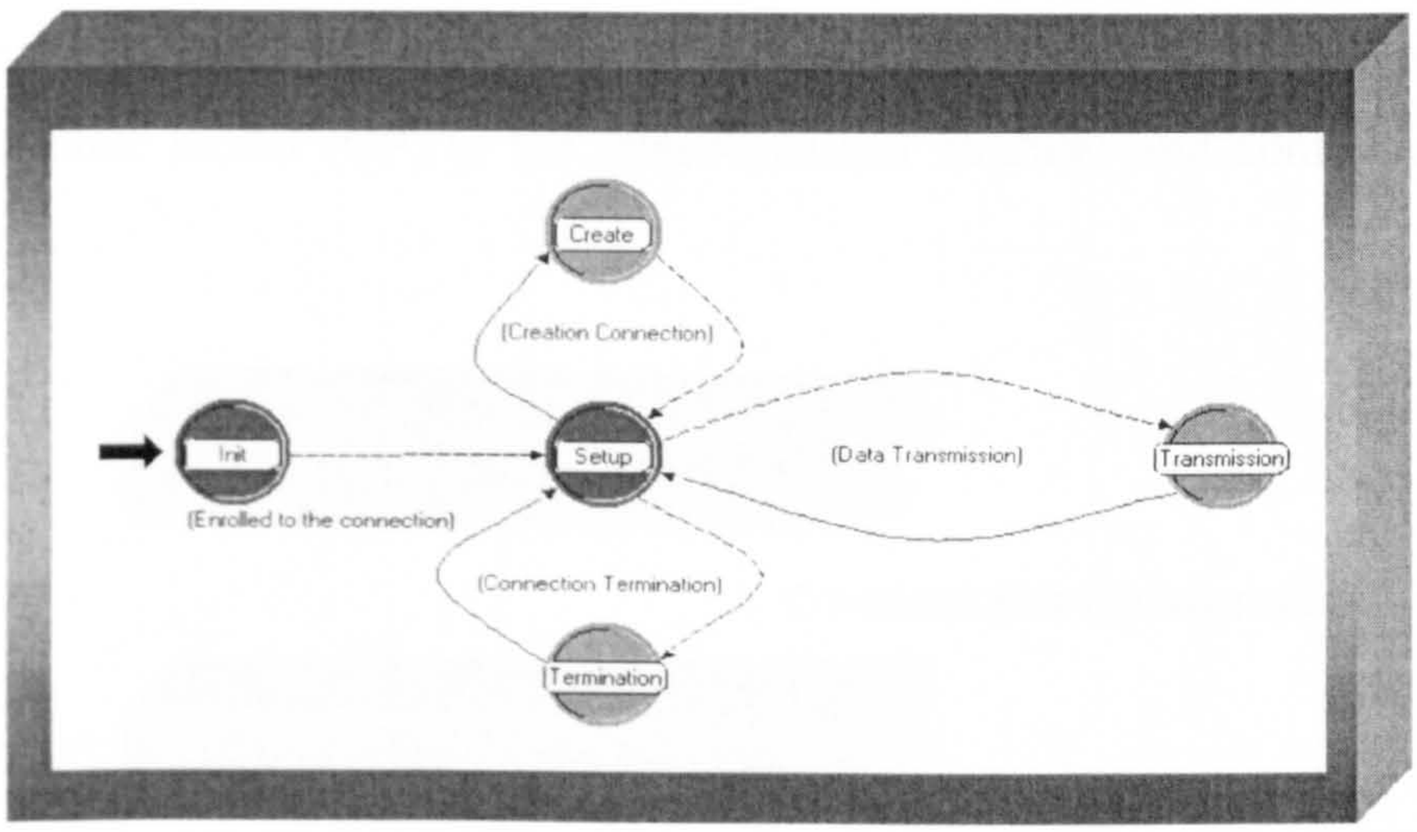


Figure (6.5): ectp_manager process model

6.2.4 "ectp_connection "STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

The following sections show how to build a state transition for ectp_connection. It illustrates the ectp_connection process model, which represents a child process within ectp_ manger to manage activities related to individual connection processes.

- **STATE TRANSITION DIAGRAM**

STAGE 1) CONTEXT DEFINITION

- **The interdependent modules of an ECTP connection are as follows:**
  1) Connection creation process;
  2) Data transmission process;
  3) Connection termination process;

- **Selecting communication mechanisms:**

  In this stage we uses all the packet communication mechanisms. Figure (6.6) defines a node model (QP) to the interdependent modules and communication transitions.

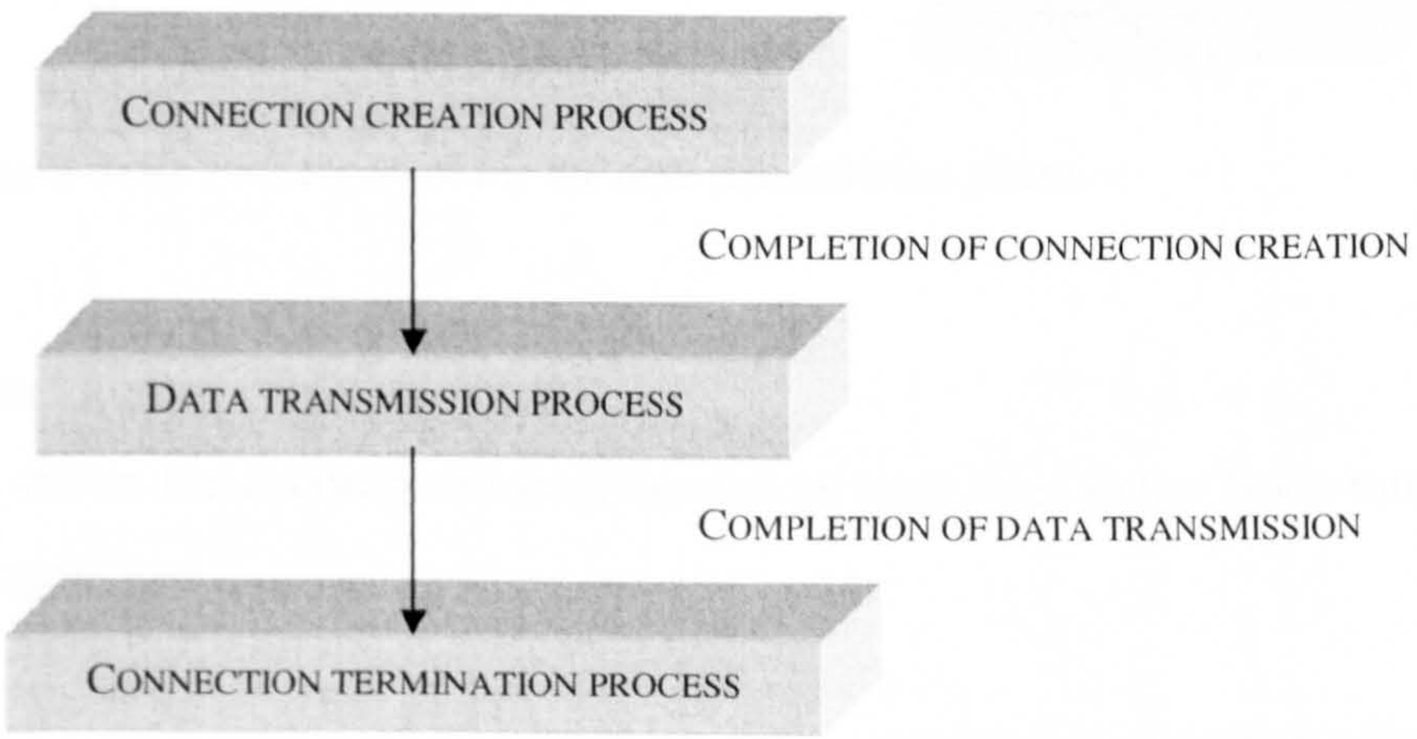


Figure (6.6): Node model for ECTP transport protocol

STAGE 2) PROCESS-LEVEL DECOMPOSITION

- **CONNECTION CREATION PROCESS**

  Figure (6.7) shows the decomposition of first process level in the Node model (previous stage).

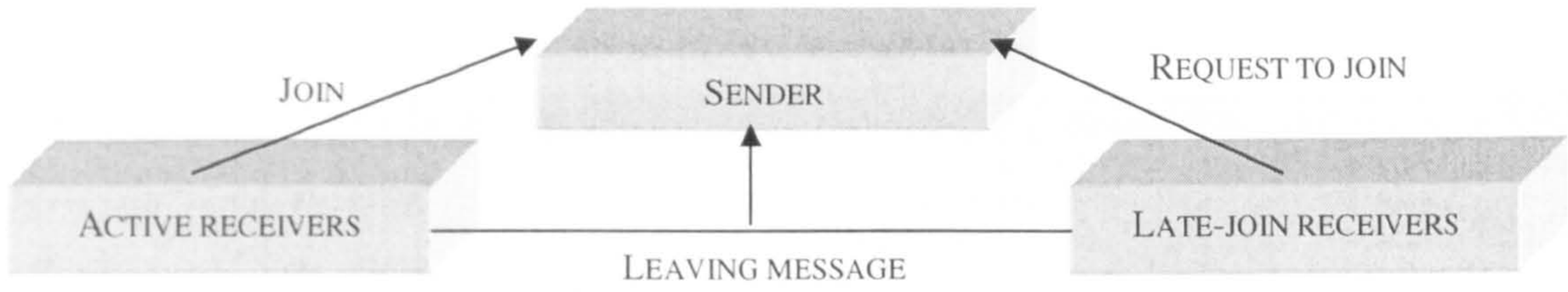


Figure (6.7): Process level for the connection creation phase

- **DATA TRANSMISSION PROCESS**

  Figure (6.8) shows the decomposition of second process level in the Node model (previous stage).

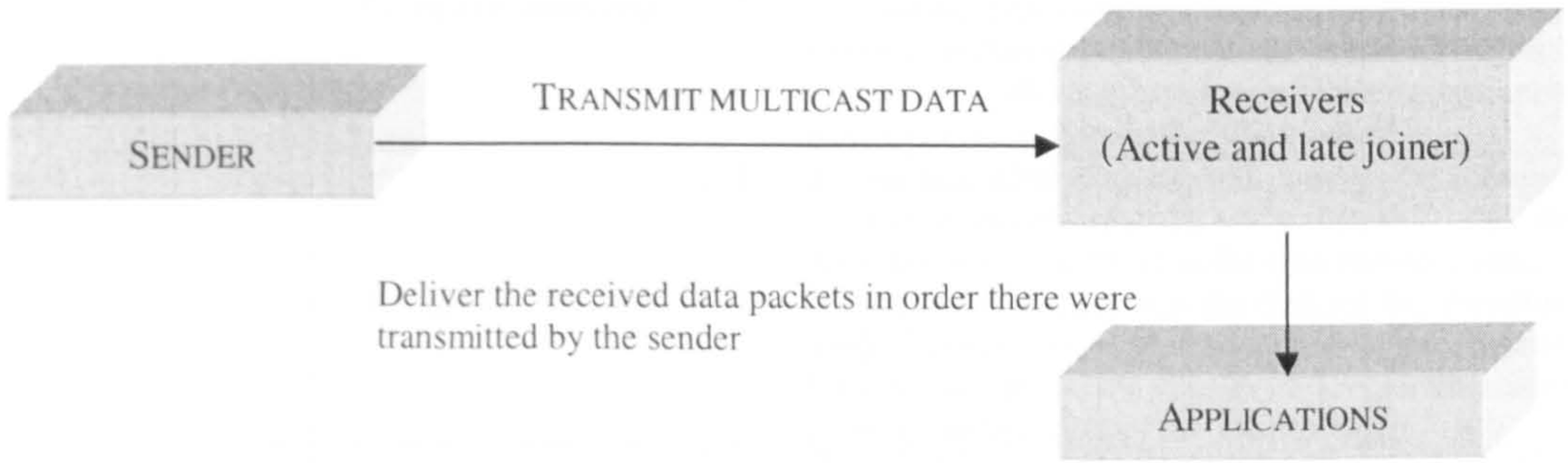


Figure (6.8): Process level for the data transmission phase

- **CONNECTION TERMINATION PROCESS**

  Figure (6.9) shows the decomposition of second process level in the Node model (previous stage).

Figure (6.9): Process level for the connection termination process

STAGE 3) ENUMERATION (DETAILS) OF EVENTS (PER PROCESS)

Table (6.7) shows connection creation process.

SERVICES	LOGICAL EVENTS	PARAMETERS
TC CONNECTION	T-CREATE REQUEST	1. A TAP address that identifies the T-user(s) expected to participate in TC being established; 2. A TAP address of TC-owner by whom the TC creation has been requested; 3. TC characteristics conveys the QoS for TC, therefore the QoS in the request primitive is what proposed by the TC-owner; 4. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-CREATE INDICATION	1. A TAP address that identifies the T-user(s) expected to participate in TC being established; 2. A TAP address of TC-owner by whom the TC creation has been requested; 3. TC characteristics conveys the QoS for TC, therefore the QoS in the indication primitive is what modified by TS-provider; 4. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-CREATE RESPONSE	1. Responding address parameters conveys the address of TAP of the TS-user to participate in TC and to which TS-user data should be delivered when the TC is in the data transfer event; 2. TC characteristics conveys the QoS for TC, therefore the QoS in the response primitive is what counter-proposed responding TS-users; 3. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-CREATE CONFIRM	1. Responding address parameters conveys the address of TAP of the TS-user to participate in TC and to which TS-user data should be delivered when the TC is in the data transfer event; 2. TC characteristics conveys the QoS for TC, therefore the QoS in the confirm primitives is what arbitrated by the TS-provider; 3. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
TC INVITATION	T-INVITE REQUEST	1. To establishing a TC, the TAP address that identifies the TS-user being invited; 2. TAP address of the TC-owner by whom the TC invitation has been requested; 3. TC characteristic parameters conveys the QoS for TC; 4. TS-user data parameter allows the transfer of the TC-owner data to other TS-user;
	T-INVITE INDICATION	1. To establishing a TC, the TAP address that identifies the TS-user being invited; 2. TAP address of the TC-owner by whom the TC invitation has been requested; 3. TC characteristic parameters conveys the QoS for TC; 4. TS-user data parameter allows the transfer of the TC-owner data to other TS-user;
TC JOIN	T-JOIN REQUEST	1. TAP address for active receivers expected to participate in TC being established. In the case of late join wherein a TS-user attempts to join an existing TC, therefore TAP conveys the group address of the TC to join; 2. In the case of late join, TAP conveys the address of the TS-user attempting to join the existing TC; 3. TC characteristics conveys the QoS for TC, therefore the QoS in the request primitive is what proposed by the TS-user; 4. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-JOIN INDICATION	1. TAP address for active receivers expected to participate in TC being established. In the case of late join wherein a TS-user attempts to join an existing TC, therefore TAP conveys the group address of the TC to join; 2. In the case of late join, TAP conveys the address of the TS-user attempting to join the existing TC;

		3. TC characteristics conveys the QoS for TC, therefore the QoS in the indication primitive is what modified by TS-provider; 4. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-JOIN RESPONSE	1. The responding address parameter conveys the address of the TAP of the TS-user to participate in the TC and to which TS-user data should be delivered when the TC is in the data transfer state. In case of a late join, the responding address conveys the address of the TC-owner. 2. TC characteristics conveys the QoS for TC, therefore the QoS in the response primitive is what counter-proposed responding TS-users; 3. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;
	T-JOIN CONFIRM	1. The responding address parameter conveys the address of the TAP of the TS-user to participate in the TC and to which TS-user data should be delivered when the TC is in the data transfer state. In case of a late join, the responding address conveys the address of the TC-owner. 2. TC characteristics conveys the QoS for TC, therefore the QoS in the confirm primitives is what arbitrated by the TS-provider; 3. TS-user data allows the transfer of TS user data among TS-users without modification by the TS-provider;

TC= Transport Connection
TAP= Transport access point
TS= Transport service

Table (6.7): Connection creation process

Table (6.8) shows Data transmission process.

SERVICES	LOGICAL EVENTS	PARAMETERS
DATA TRANSFER	**T-DATA REQUEST**	1. TS-user data parameter allows the transfer of data from a sending TS-user to the receiving TS-user(s), without modification by the TS-provider;
	T-DATA INDICATION	1. TAP conveys an address that identifies the TS-user who has sent the data; 2. Status parameter notifies to detect but not correct errors which is signalled to the TS-user; 3. TS-user data parameter allows the transfer of data from a sending TS-user to the receiving TS-user(s), without modification by the TS-provider;
	T-UNITDATA REQUEST	1. TAP conveys an address that identifies the TS-user expected to received the data sent, (this parameter may be present only in this event); 2. TAP conveys an address that identifies the TS-user who has sent the data; 3. All parameters of TC characteristics expect the transit delay to be null value,(this parameter may be present only in this event); 4. TS-user data parameter allows the transfer of data from a sending TS-user to the receiving TS-user(s), without modification by the TS-provider;
	T-UNITEDATA INDICATION	1. TAP conveys an address that identifies the TS-user expected to received the data sent, (this parameter may be present only in this event); 2. TAP conveys an address that identifies the TS-user who has sent the data; 3. All parameters of TC characteristics expect the transit delay to be null value,(this parameter may be present only in this event); 4. Status parameter notifies to detect but not correct errors which is signalled to the TS-user; 5. TS-user data parameter allows the transfer of data from a sending TS-user to the receiving TS-user(s), without modification by the TS-provider;

PAUSE	**T-PAUSE INDICATION**	The parameter gives information indicating the cause of the data transfer suspension. The reason is one of the following: 1. Temporary lack of local or remote resources at TC-provider; 2. A QoS parameter temporarily below an agreed LQA level (QoS negotiation level);
RESUME	**T-RESUME INDICATION**	The parameter gives information indicating the reason for the resumption of the data transfer. The reason may usually be the recovery from the bottleneck that caused by the previous Pause service;
REPORT	**T-REPORT INDICATION**	The parameter gives information indicating the cause of the report. The reason is one of the following: 1. Minor lack of local or remote resources at TC-provider; 2. Detected but not fatal QoS change;

TC= Transport Connection
TAP= Transport access point
TS= Transport service

Table (6.8): Data transmission process

Table (6.9) shows Connection termination process.

SERVICES	LOGICAL EVENTS	PARAMETERS
TC LEAVE	**T-LEAVE REQUEST**	1. The parameter conveys: a) in the T-leave request a group of TAP address that identifies the TC to leave; b) in the T-leave indication, the TAP address of the TS-user would be excluded from the TC; 2. TAP conveys an address of the TS-user to be excluded from the TC;
	T-LEAVE INDICATION	1. The parameter conveys: c) in the T-leave request a group of TAP address that identifies the TC to leave; d) in the T-leave indication, the TAP address of the TS-user would be excluded from the TC; 2. The parameter givers information on the cause of the leave. The reason is one of the following: a) QoS parameter below an agreed LQA level (QoS negotiation); b) Lack of local or remote resources at the TC provider;
TC TERMINATION	**T-TERMINATION REQUEST**	The TS-user data parameter is present in the T-termination request when a termination request was originated by the TC-owner;
	T-TERMINATION INDICATION	1. The parameter gives information regarding the cause of the termination. The reason is one of the following: a) TC-owner invoked termination; b) Lack of local or remote resource at TC provider; c) QoS below an agreed LQA level (QoS negotiation); 2. The TS-user data parameter is present in the T-termination request when a termination request was originated by the TC-owner;

TC= Transport Connection
TAP= Transport access point
TS= Transport service

Table (6.9): Connection termination process

STAGE 4) STATE-LEVEL DECOMPOSITION (PER PROCESS)

Table (6.10) describes the description of each state in the each process.

STATES	DESCRIPTIONS
IDLE	Waiting for potential receivers to enrolled onto the multicast group,
INVITE PENDING	Enrolled receivers await to receive a connection creation message from the ECTP sender and responses back with confirmation message to sender,
INCOMING JOIN PENDING	Connection creation completed when the sender receives the confirmation messages from all the active receivers,
OUTGOING JOIN PENDING	The receivers that have joined the connection are active receivers, The active receivers can leave the connection by sending a leaving request to the sender,
OUTGOING JOIN RESPONSE PENDING	The late joiner receivers send a join request to the sender the sender transmits a join confirm message, which indicates whether the join request is accepted or not, The late joiner receivers can leave the connection by sending a leaving request to the sender,
OUTGOING CREATE PENDING	ECTP transport connection is created for the enrolled receivers,
DATA TRANSFER READY	The sender begins to transmit multicast data
DATA TRANSFER SUSPEND	During data transmission, if server network congestion is indicate by the QoS management function, the sender suspends the multicast data transmission temporality,
INCOMING JOIN PROCESSING (1)	The sender begins to transmit data transmission after the period of data transmission suspension, The sender sends a termination message to all the receivers,
INCOMING JOIN PROCESSING (2)	The receivers deliver the received data packets to the applications in the same order the sender transmitted them, The sender sends a termination message to all the receivers,

Table (6.10): Description of each state in the each process

STAGE 5) STD DEVELOPMENT (PER PROCESS)

Table (6.11) shows the Event response for all processes.

STATE	LOGICAL EVENTS	NEXT STATE
IDLE	T-INVITE.REQ T-CREAT.REQ	INVITE PENDING OUTGOING CREATE PENDING
INVITE PENDING	T-JOIN.IND T-JOIN.REQ	INCOMING JOIN PENDING OUTGOING JOIN PENDING
INCOMING JOIN PENDING	T-LEAVE.REQ T.JOIN.RES	INVITE PENDING OUTGOING JOIN RESPONSE PENDING
OUTGOING JOIN PENDING	T-LEAVE.REQ T-LEAVE.IND T-JOIN.CON	INVITE PENDING DATA TRANSFER READY
OUTGOING JOIN RESPONSE PENDING	T-LEAVE.REQ T-LEAVE.IND T-REPORT.IND	INVITE PENDING DATA TRANSFER READY
OUTGOING CREATE PENDING	T-CREATE.CON	DATA TRANSFER READY
DATA TRANSFER READY	T-PAUSE.IND T-JOIN.IND T-DATA.REQ,IND, T-REPORT.IND T-INVITE.REQ	INCOMING JOIN PROCESSING (2) DATA TRANSFER SUSPEND
DATA TRANSFER SUSPENDED	T-RESUME.IND T-JOIN.IND T-REPORT.IND	DATA TRANSFER READY INCOMING JOIN PROCESSING (1)
INCOMING JOIN PROCESSING1	T-JOIN.RES T-LEAVE.REQ T-REPORT.IND	DATA TRANSFER SUSPEND
INCOMING JOIN PROCESSING 2	T-JOIN.RES T-LEAVE.REQ T-DATA.REQ,.IND, T-REPORT.IND	DATA TRANSFER READY

Table (6.11): Event response for all processes

- **ectp_connection Process model**

Figure (6.10) illustrates the process model of ectp_manager and their states' transitions in OPNET. The ectp_connection process model represents a child process within ectp_ manger to manage activities, related to individual connection processes.

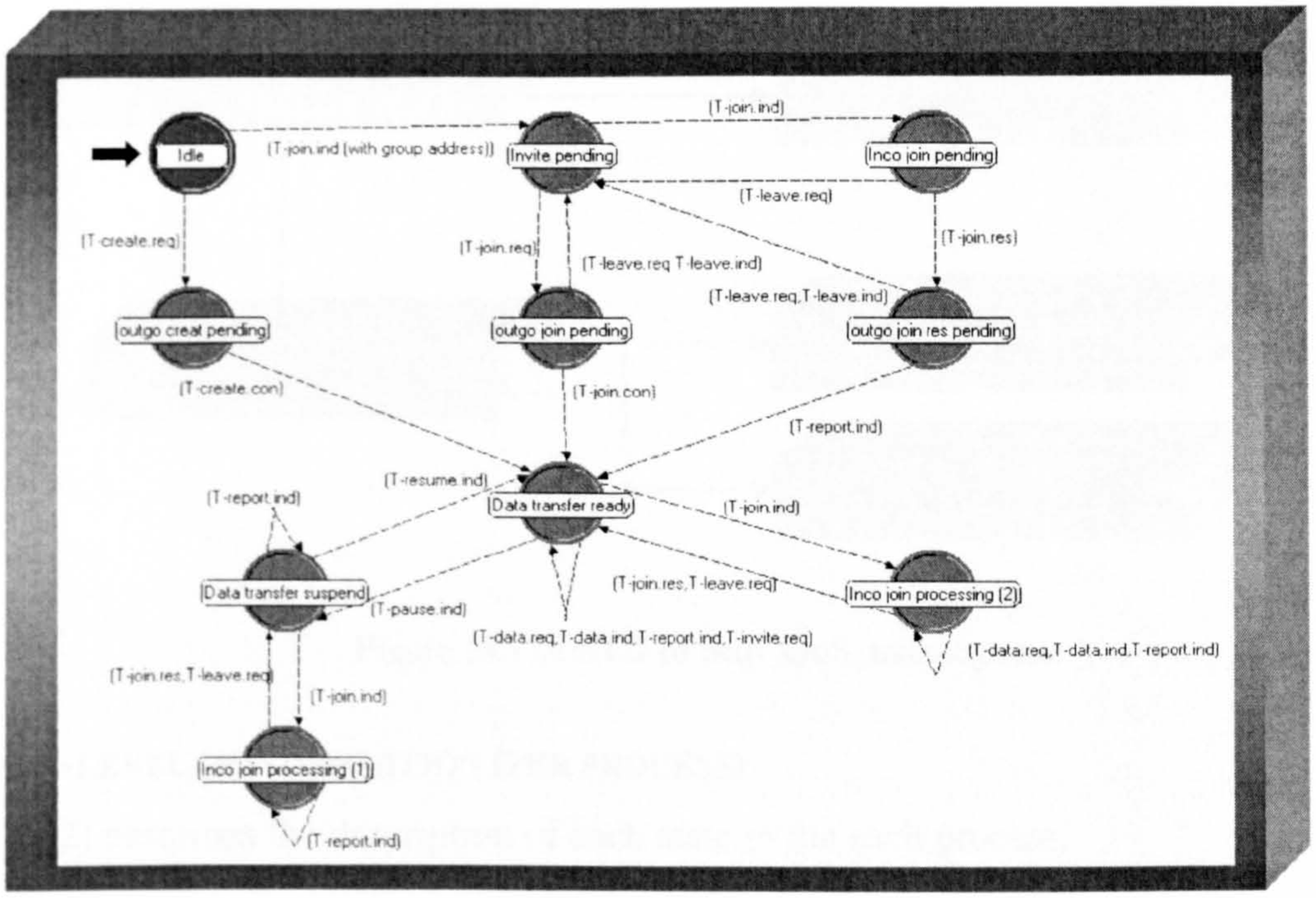


Figure (6.10): ectp_connection process model

6.2.5 "ectp_QoS_management "STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

Figure (6.11) illustrates a state transition diagram (STD) for ectp_QoS_management and defines the interdependent modules and communication transitions.

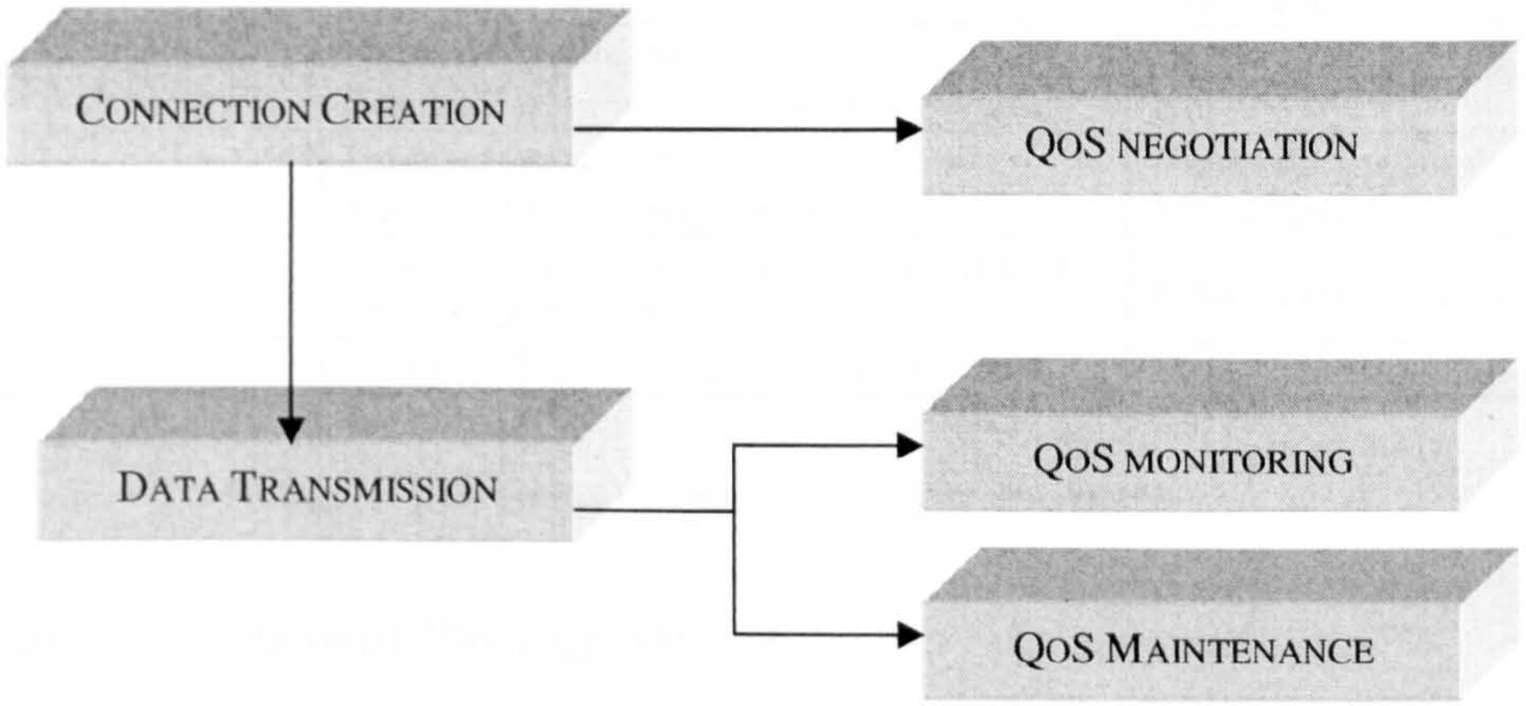


Figure (6.11): STD of ectp_QoS_manaegmnet

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.12) describes the description of each state in the each process.

STATE	DESCRIPTION
Init	Initials the connection creation phase
Connection_ Creation	Sending the CR packet to the all receivers to start the connection.
Data_ Transmission	Monitoring and Maintenance performed in the data transmission phase.
Negotiation	The ECTP sender transmits a CR packet to all receivers. The receiver responses to sender with a CC packet containing the modified target value.
Responds	Each receiver can refer a new "modified value" in response to a target value proposed by sender.
Proposed	ECTP sender proposes the desired target values such as CHQ, OT, LQA, for each QoS parameter to all receivers.
Monitoring	QoS monitoring provides the sender with the information about how well the connection is operating. Sender maintains the connection status at a desired QoS level.
Maintenance	QoS maintenance is performed to maintain the quality of a connection at a desired level and to prevent the connection quality from being degraded below the negotiated QoS level.

Table (6.12): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.13) shows the Event response for all processes.

STATE	LOGICAL EVENTS	NEXT STATE
Init	Sender/Start by CR packets	Connection_ Creation
Connection_ creation	CR packet ECTP connection is enabled	Negotiation Data Transmission
Negotiation	CC Packet	Connection_ Creation
Responds	Receiver/Modified value	Negotiation
Proposed	Sender/Target value parameter	Negotiation
Data_ Transmission	Enabled QoS management Enabled QoS management	Monitoring Maintenance
Monitoring	Sender/Maintain the connection status at a desired QoS level	Data_ Transmission
Maintenance	Receiver/Measure the parameter values	Data_ Transmission

Table (6.13): Event response for all processes

- **ectp_QoS_management PROCESS MODEL**

Figure (6.12) illustrates the process model of ectp_QoS_management and their states' transitions in OPNET. ectp_QoS_management represents a child process within ectp_ manger to manage QoS level through out the connection and transmission processes.

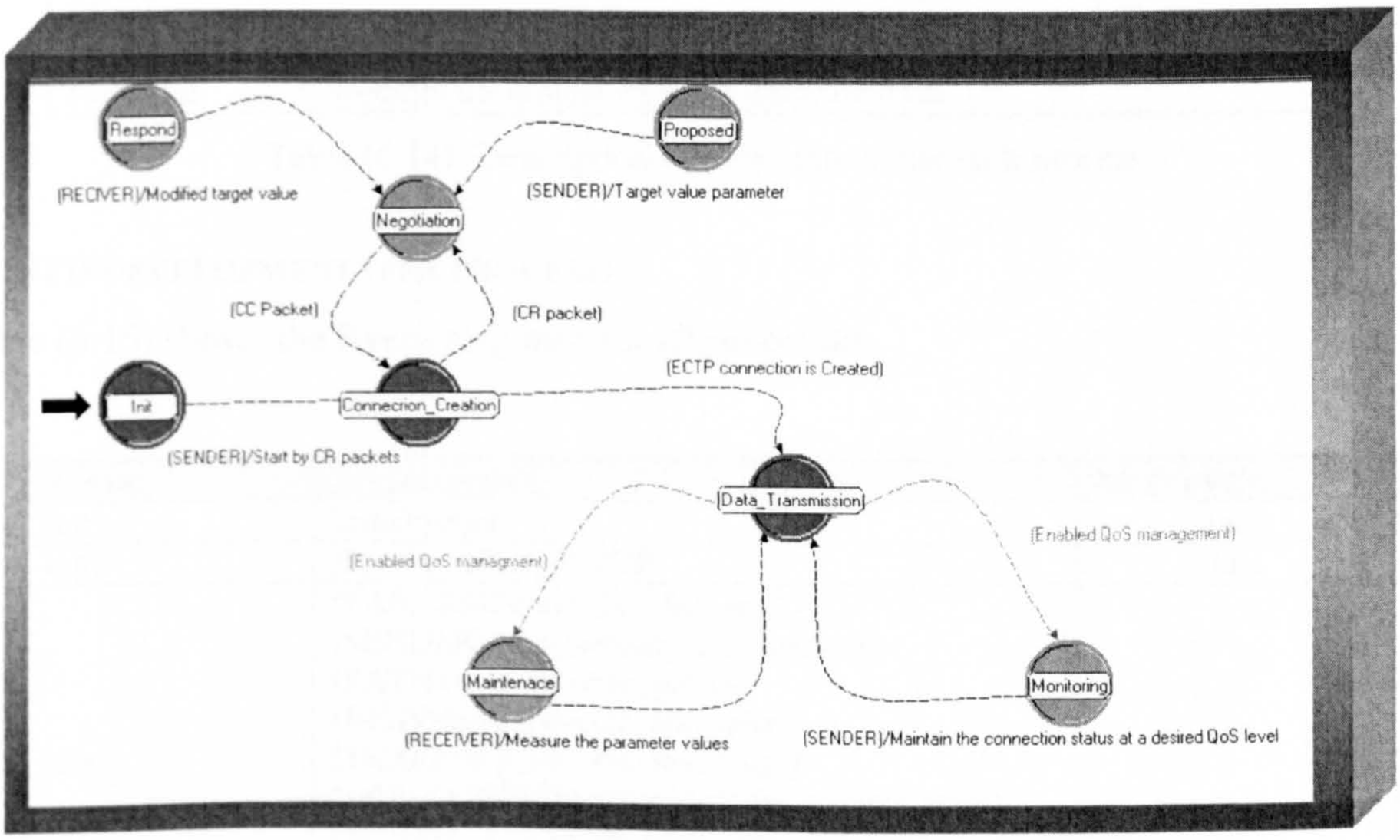


Figure (6.12): ectp_QoS_management process model

6.2.6 "rsvp" STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

rsvp process model has been pre-defined in the OPNET library. Figure (6.13) illustrates a state transition diagram (STD) for rsvp and defines the interdependent modules and communication transitions.

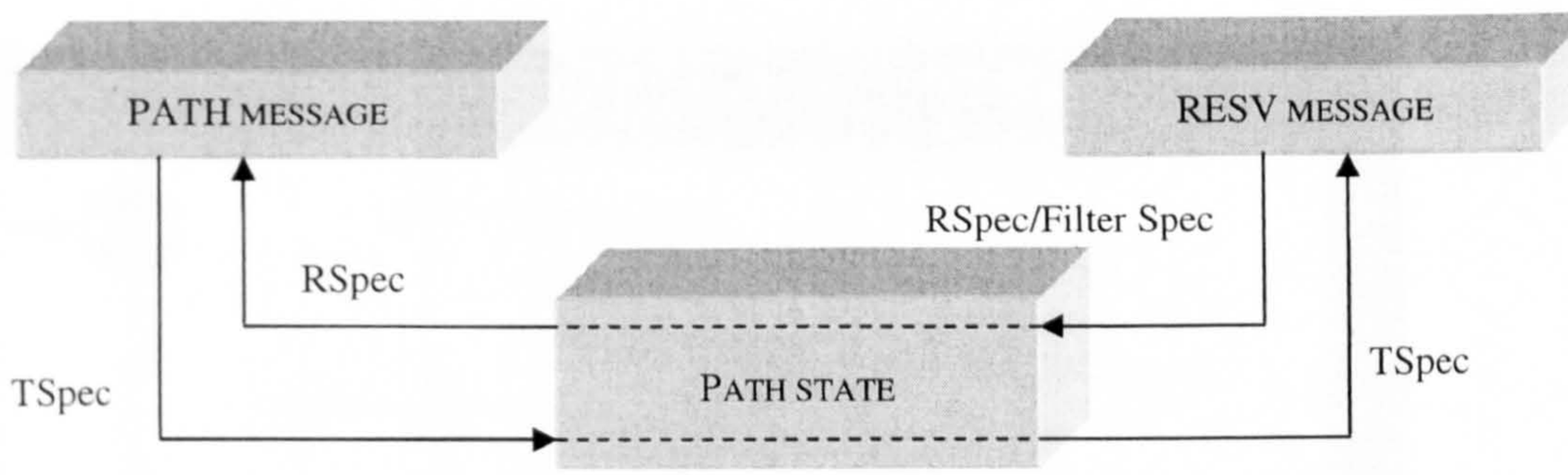


Figure (6.13): rsvp State transition diagram (STD)

The modules of the rsvp are defined as follow:

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.14) shows the description of each state in the each process.

STATE	DESCRIPTION
PATH message	Contains the traffic specification (TSpec) information to the destination address.
RESV message	Contains the request specification (RSpec) that indicates the type of Intserv and the filter specification that characterises the packets for which the reservation is being made.
Path state	Specifies the downstream and upstream route.

Table (6.14): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.15) shows the Event response for all processes.

STATE	LOCAL EVENT	NEXT EVENT
Init1	Initialisation	Init2
Init2	IP_OR_MPLS_NOTIF	Idle
Idle	(FAIL_REC)/rsvp_Isp_fail_rec() (SENDER)/rsvp_sender_start_session() (PATH)/rsvp_process_path() (RESV)/rsvp_process_resv_msg() (TEAR)/rsvp_process_tear_msg() (ERR_CONFIRM)/rsvp_process_err_con_msg() (SIGNAL_LSP)/rsvp_send_label_request() (SEND_LABEL)/revp_send_label() (REFRESH)/rsvp_send_refresh_msg()	

Table (6.15): Event response for all processes

- **rsvp PROCESS MODEL**

Figure (6.14) illustrates the process model of rsvp and their states' transitions in OPNET. rsvp represents a child process within ectp_QoS_management process model to manage QoS level through out the connection and transmission processes.

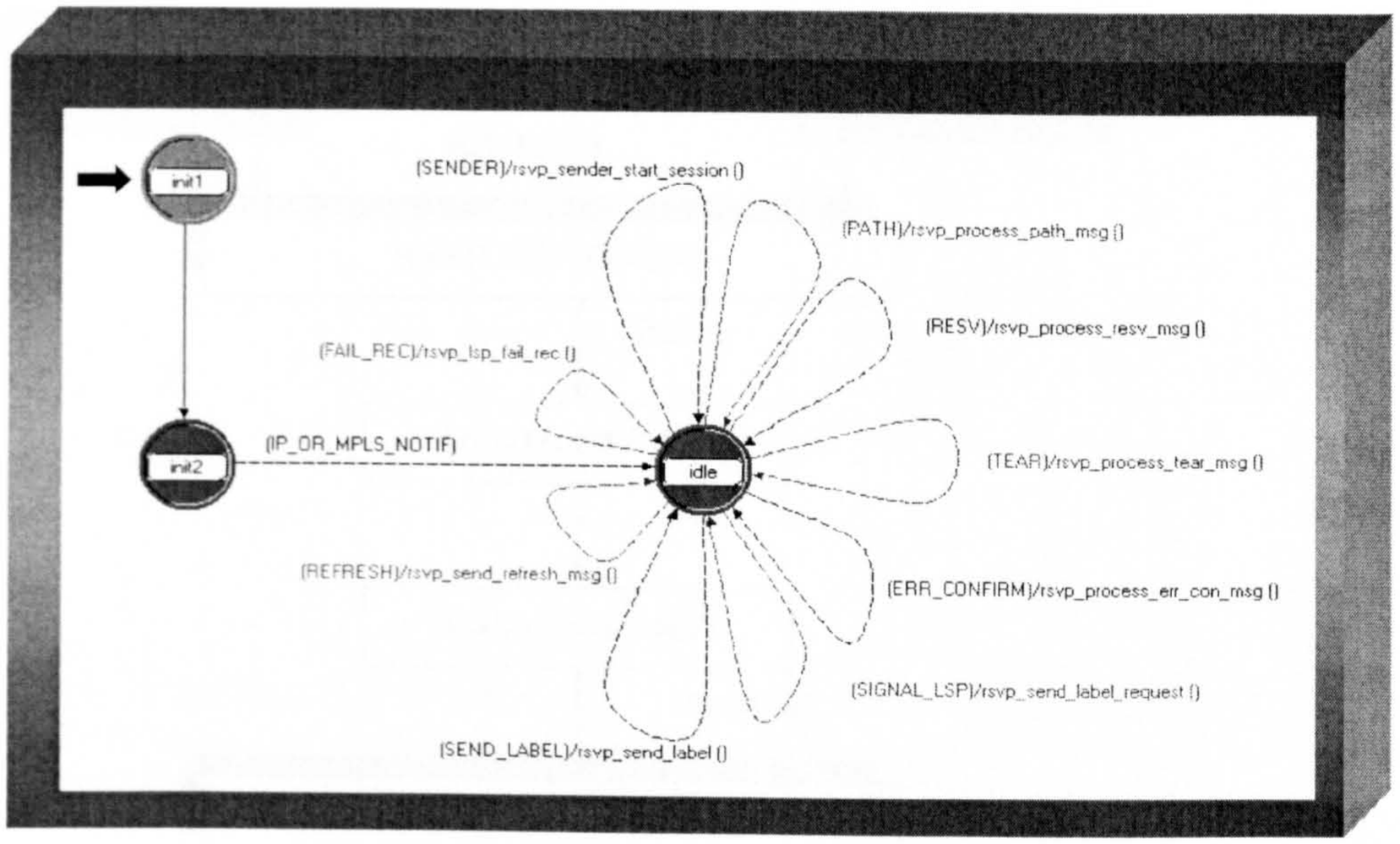


Figure (6.14): rsvp process model

6.2.7 "aggregated_rsvp" STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

Figure (6.15) illustrates a state transition diagram (STD) for aggregated_rsvp and defines the interdependent modules and communication transitions.

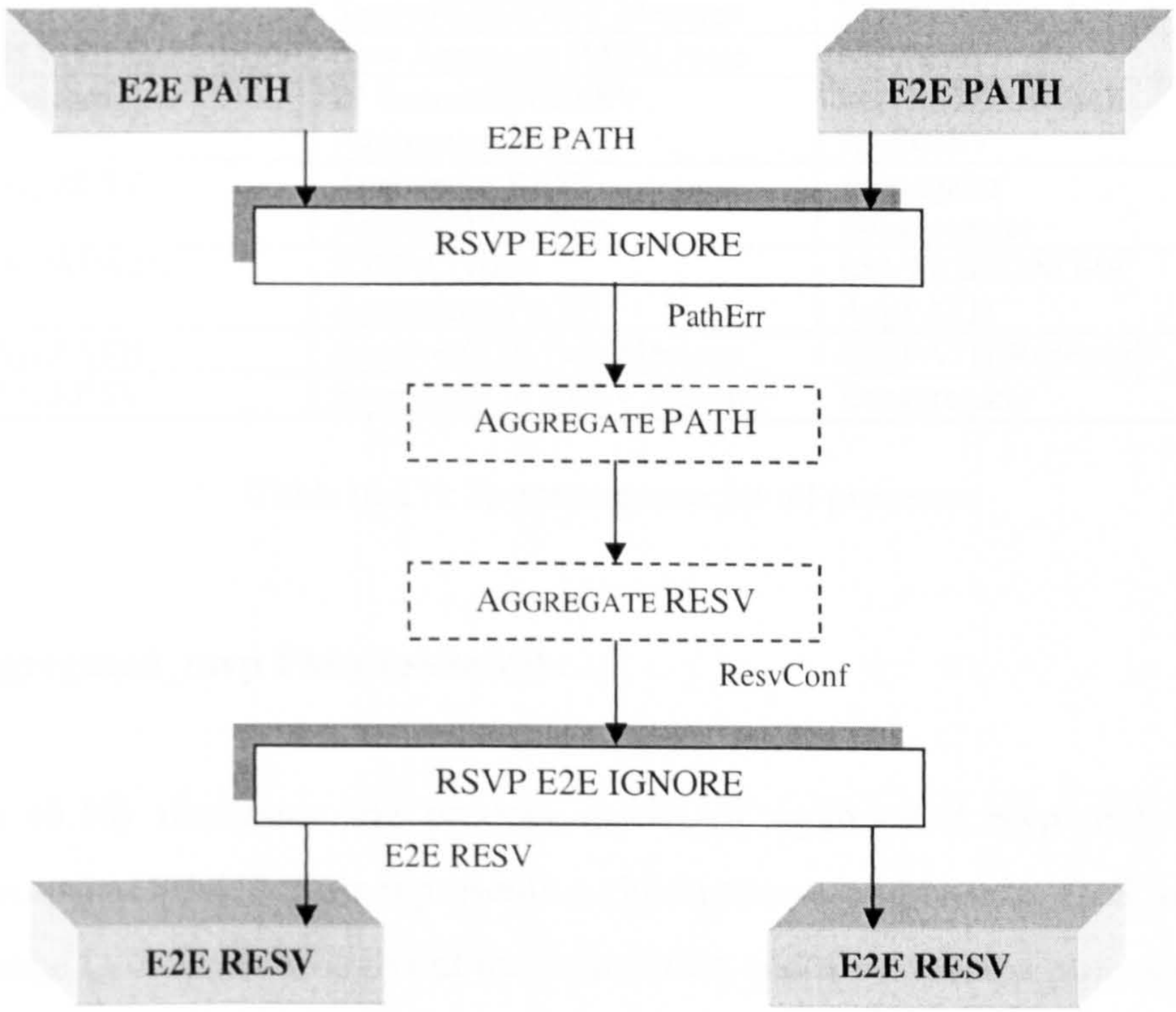


Figure (6.15): Aggregated RSVP State transition diagram (STD)

The modules of the aggrsvp can be defined as follow:

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

  Table (6.16) describes the description of each state in the each process.

STATE	DESCRIPTION
E2E PATH	Aggregator forwards E2E Path into aggregation region
RSVP E2E IGNORE	RSVP end-to-end ignore region or IP tunnel
Aggregate Path	Aggregator signals an Aggregator PATH
Aggregate RESV	Deaggregator signal Aggregator RESV
E2E RESV	Deaggregator forwards E2E RESV into aggregation region

Table (6.16): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.17) shows the Event response for all processes.

STATE	LOCAL EVENT	NEXT EVENT
init	initialisation	E2E PATH (Sender)
E2E PATH	Sender/ E2E Path message	Aggregator
Aggregator	IP tunnel/E2E PATH AggRESVConfirm Sender/E2E RESV Message	RSVPE2EIGNORE AggRESV E2ERESV(Sender)
RSVPE2EIGNORE	New Aggregate PATH State	NEWPATH
Deaggregator	IP Tunnel/E2ERESV Aggregator RESV	RSVPE2EIGNORE AggRESV
AggRESV	Aggregator RESV AggRESVConfirm	Aggregator Deaggregator
NEWPATH	E2EPATHErr Aggregator PATH	RSVPE2EIGNORE AggPATH
AggPATH	Receiver/E2E Path Message	E2EPATH(Receiver)
E2ERESV	Receiver/E2E RESV Message	Deaggregator

Table (6.17): Event response for all processes

- **aggregated_rsvp PROCESS MODEL**

Figure (6.16) illustrates the process model of aggregated_rsvp and their states' transitions in OPNET. This represents a child process within ectp_QoS_management to manage QoS level through out the connection and transmission processes.

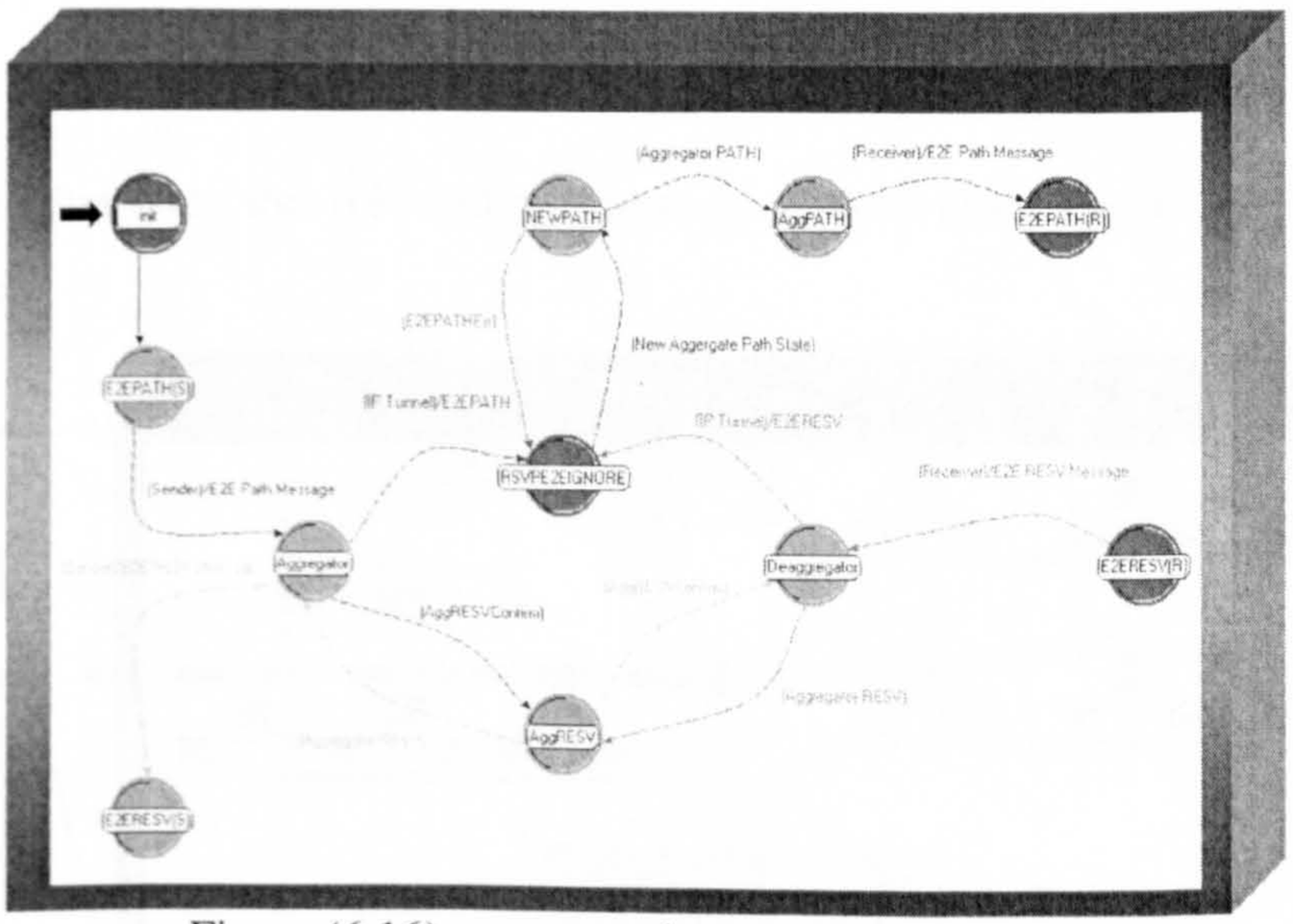


Figure (6.16): aggregated_rsvp process model

6.3 IMPLEMENTATION ON THE DS ROUTERS IN OPNET

To implement the DS scheme in a router [WN01], there exist two options to choose from: a) Implement the DS-enabled IP module from scratch; b) Modify an existing router implementation available from the OPNET model library.

In this work we choose the second option, to modify the exiting OPNET model for **Cisco 7204** router. In order to handle traffic from different classes, the IP process model had to be modified by adding the DS components.

Each node intended to use *ip* has an "ip" module. Table (6.18) shows the modified ip process models. The first process model is *DS_ip* process model, which represents the parent (root) process of the ip module. This process model is a modified of ip_dispatch in the ip module. The second one is *DS_ps* process model, which represents the child process in the ip module. The third model is ip_icmp from the OPNET process model's library, which represents the child process in the ip module.

IP PROCESS MODELS		
DS_ip	Represent the parent (root) process model of "ip" module. It manages a set ip process, which implements main IP and Diffserv functionality.	Parent process model
DS_ps	Represents priority packet scheduling by using PS queue for the premium class traffic and the RIO-queue for assured and best effort traffic.	Child process model
ip_icmp	for handling the echo requests and replies that are configured in a network.	Child process model

Table (6.18): ip process models

Figure (6.17) illustrates the relevant portions of typical node model where the ip model resides.

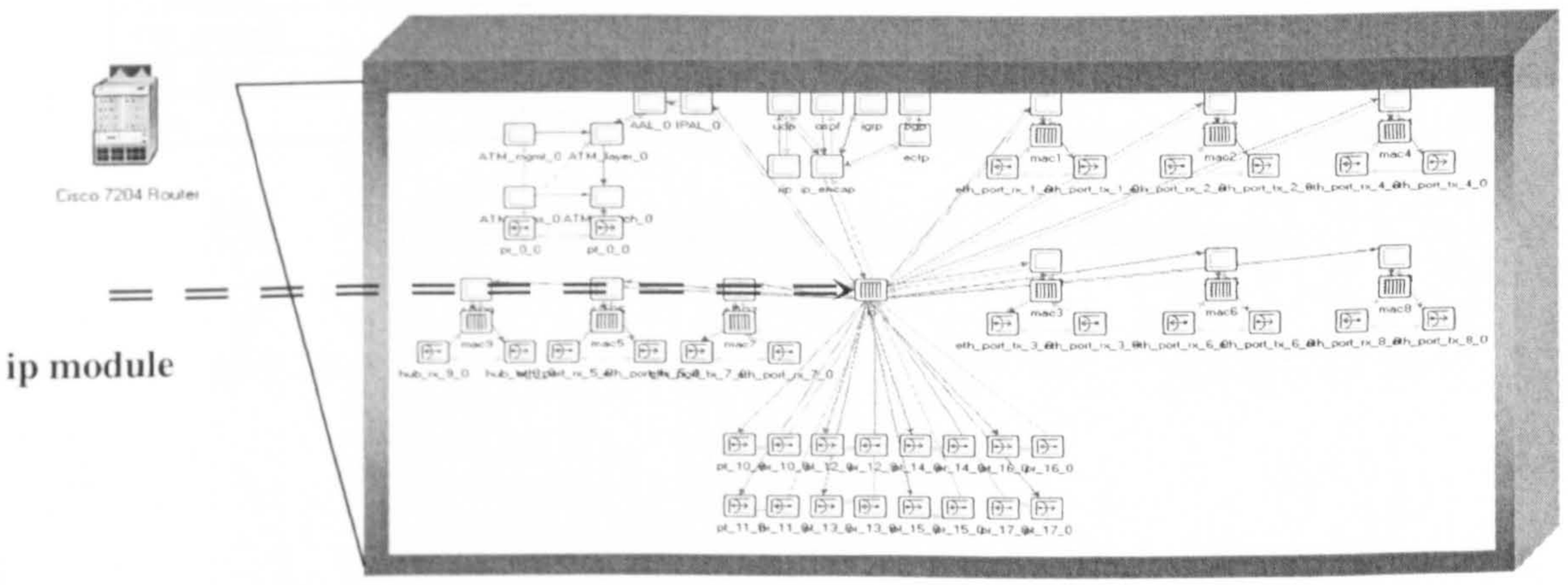


Figure (6.17): ip module in node model

6.3.1 "DS_ip" STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

DS_ip model is the main IP process, which implements the main IP and Diffserv functionality. Figure (6.18) illustrates a state transition diagram (STD) for DS_ip and defines the interdependent modules and communications transitions.

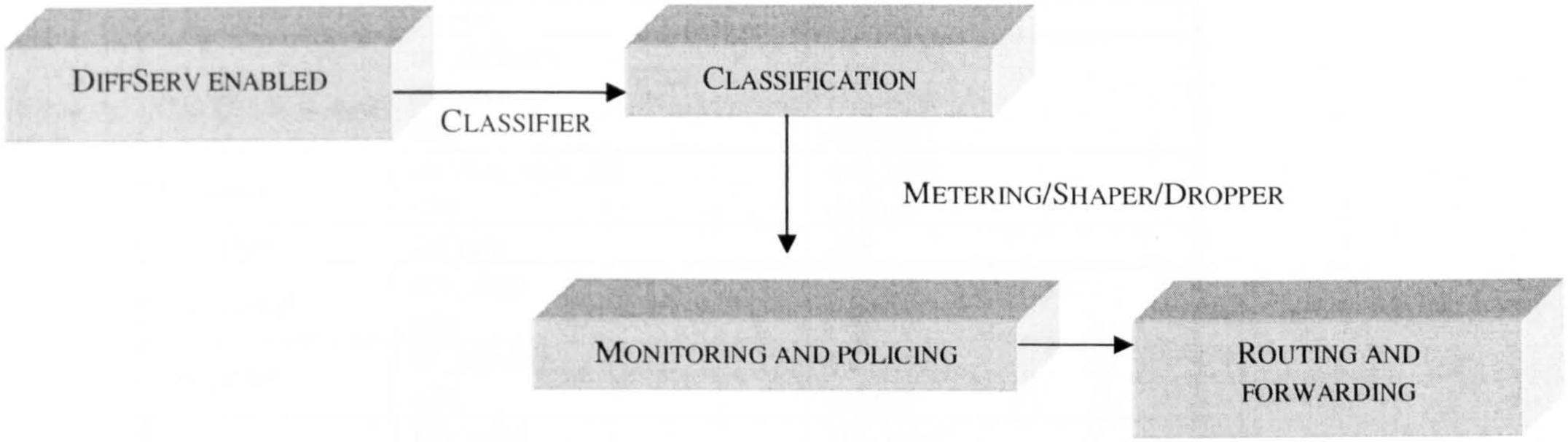


Figure (6.18): DS_ip State transition diagram (STD)

The modules of the DS_ip can be defined as follow:

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.19) describes each state in the each process.

STATE	DESCRIPTION
Init	initialisation
Wait	initialisation
Cmn_rte_tbl	To initiate the initial redistribution of routing information between the routing protocols configured on this router.
init_too	initialisation
Arrival	If the node is set to DS-enable, the transition labelled with DiffServ condition occurs.
IP_service	Packet routing and forwarding
svc_start	Packet routing and forwarding
svc_compl	Packet routing and forwarding
DS_schd	Packet classification, monitoring and policing
Idle	Packet routing and forwarding

Table (6.19): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.20) shows the event response for all processes.

STATE	LOCAL EVENTS	NEXT STATE
Init		wait
wait	self_notification	cmn_rte_tbl
cmn_rte_tbl	self_notification	init_too
init_too	arrival	arrival
arrival	no_diffserv diffserv idle	IP_servic DS_schd default
IP_servic	service_new_pk idle	svc_start default
svc_start	default	idle
svc_compl	scv_start idle	service_queued_pk default
DS_schd	IP_servic idle	PK_ready default
idle	DS_schd svc_completion arrival	DS_schd svc_compl arrival

Table (6.20): Event response for all processes

Figure (6.19) illustrates the process model of DS_ip and their states' transitions in OPNET. DS_ip represents the parent (root) process model of "ip" module. It manages a set ip process, which implements main IP and Diffserv functionality.

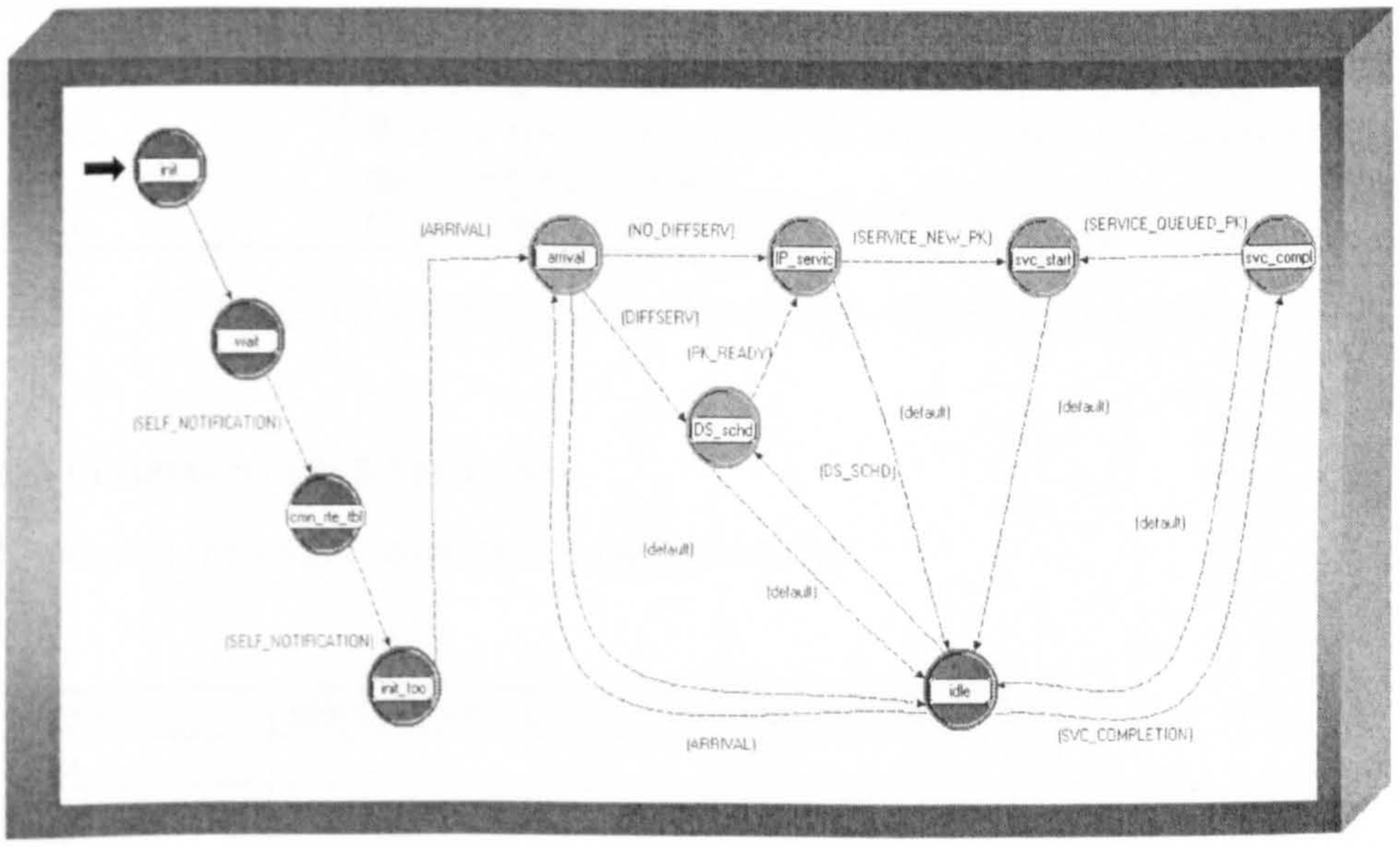


Figure (6.19): DS_ip process model

6.3.2 "DS_ps" STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

DS_ps process model handles priority packet scheduling by using two queues: the PS-queue for the premium class traffic, and the RIO-queue for assured and best-effort traffic. Figure (6.20) illustrates a state transition diagram (STD) for DS_ps and defines the interdependent modules and communications transitions.

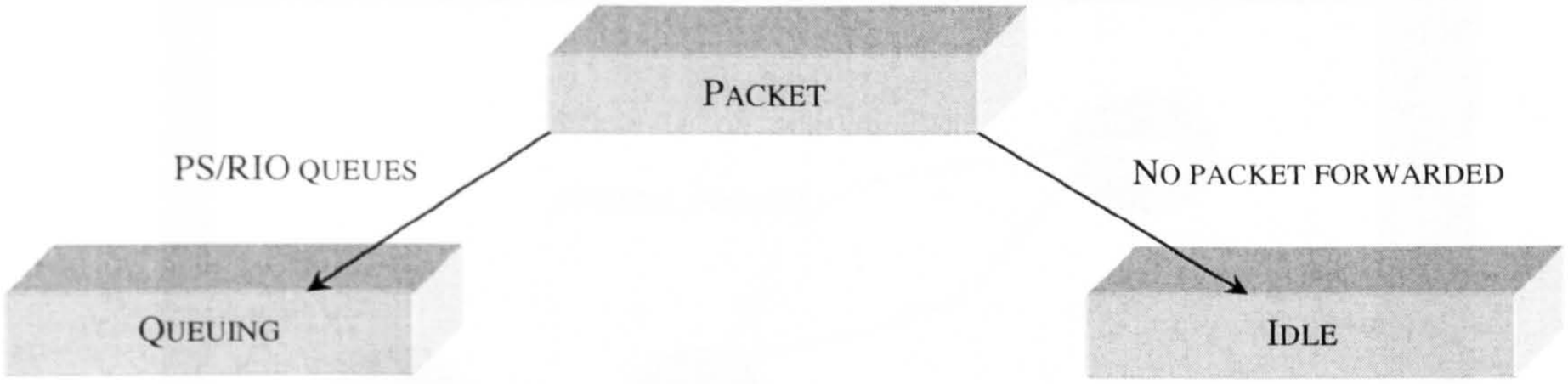


Figure (6.20): DS_ps State transition diagram (STD)

The modules of the DS_ps can be defined as follow:

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.21) describes each state in the each process.

STATE	DESCRIPTION
Init	
Idle	If no packet is coming and no packet is being scheduled.
enqueue	When a new packet enters this state where the PS-queue and RIO-queue are implemented.
extract	Checks whether there is other packet in the output queue.

Table (6.21): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.22) shows the event response for all processes.

STATE	LOCAL EVENTS	NEXT STATE
init		idle
idle	receive_ packet send_ packet	enqueue extract
enqueue	receive_ packet	idle
extract	send_ packet	idle

Table (6.22): Event response for all processes

Figure (6.21) illustrates the process model of DS_ps and their states' transitions in OPNET. DS_ps represents the child process model of DS_ip process model. It manages the priority packet scheduling by using PS queue for the premium class traffic and the RIO-queue for assured and best effort traffic.

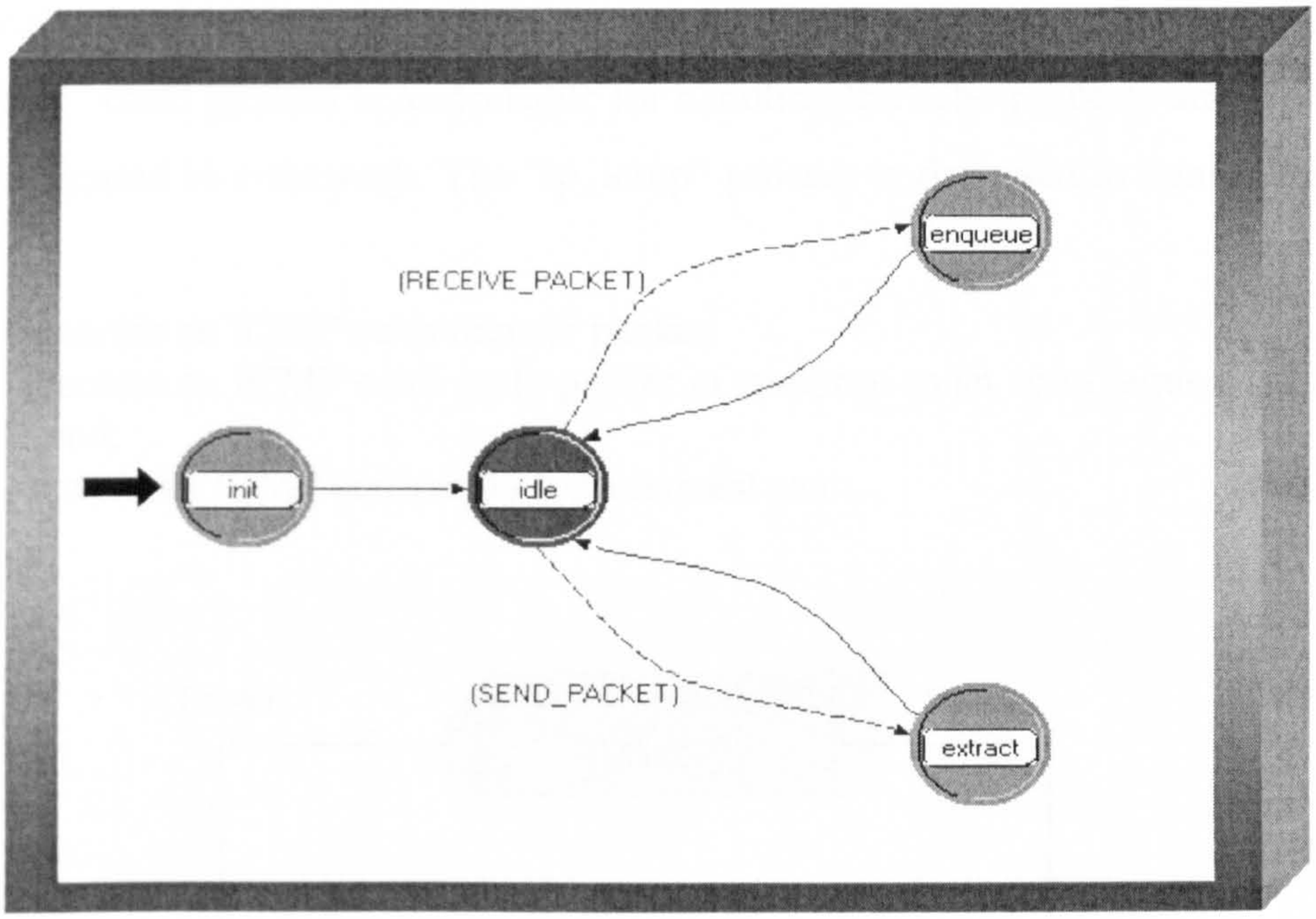


Figure (6.21): DS_ps process model

6.3.3 "ip_icmp" STATE TRANSITION DIAGRAM AND THE PROCESS MODEL

Internet Control Message Protocol (ICMP), [PostelJ81] is a required protocol tightly integrated with IP. ICMP messages, delivered in IP packets, are used for out-of-band messages related to network operation. Figure (6.22) illustrates a state transition diagram (STD) for ip_icmp and defines the interdependent modules and communications transitions.

The "ip_icmp" child process is responsible for handling the echo requests and replies that are configured in a network. The "ip_icmp" process is designed to handle three processes:

1. Generate an ICMP echo request packet.
2. Generate an ICMP echo reply packet in response to an echo request packet received.
3. Receive an ICMP echo reply for a request sent.

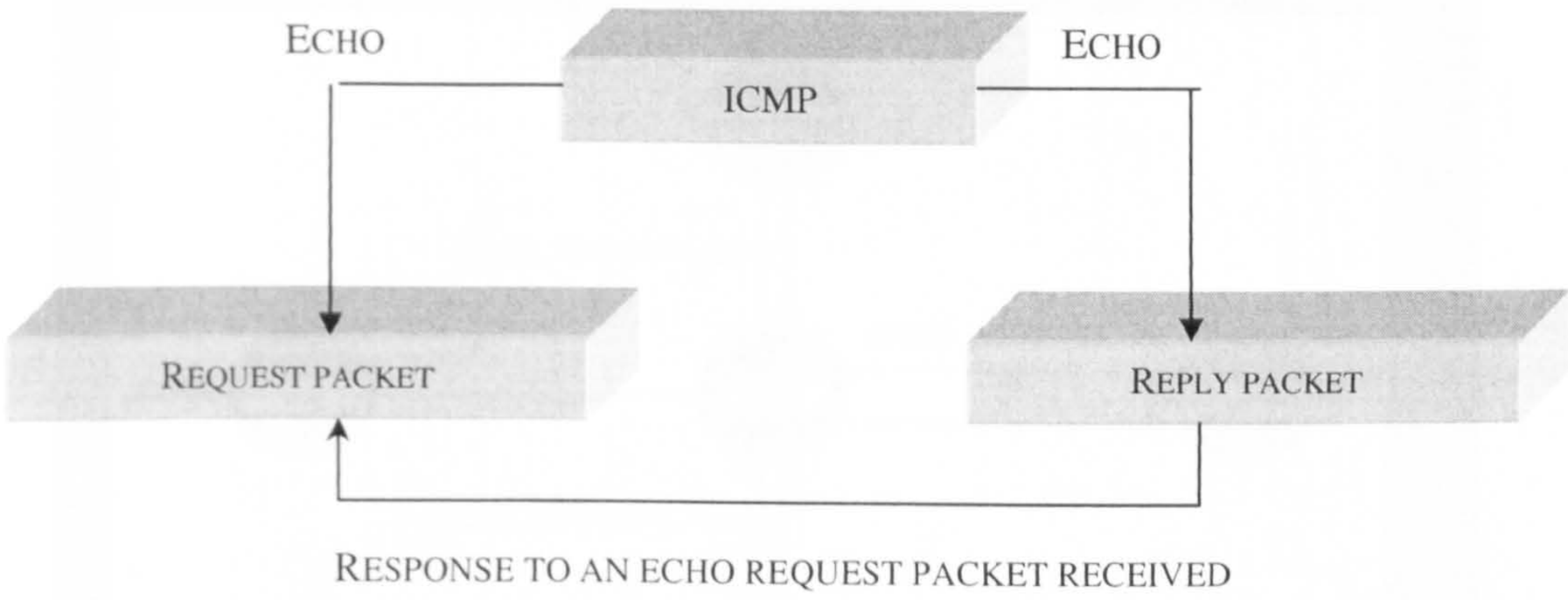


Figure (6.22): ip_icmp State transition diagram (STD)

The modules of the DS_ps can be defined as follow:

- **STATE-LEVEL DECOMPOSITION (PER PROCESS)**

Table (6.23) describes each state in the each process.

STATE	DESCRIPTION
Init	initializing all elements
Wait	Waits for a new ICMP ping request to be generated
request_gen	a new ICMP Echo request to be sent to the specified destination.
reply_ received	handles the ICMP echo reply traffic received by this node.
request_rcvd	handles the ICMP echo request traffic received by this node.

Table (6.23): Description of each state in the each process

- **STD DEVELOPMENT (PER PROCESS)**

Table (6.24) shows the event response for all processes.

STATE	LOCAL EVENTS	NEXT STATE
init		wait
wait	echo_request_gen echo_request_rcvd echo_reply_rcvd	request_gen request_rcvd reply_received
request_gen	echo_request_gen	wait
reply_received	echo_reply_rcvd	wait
request_rcvd	echo_request_rcvd	wait

Table (6.24): Event response for all processes

Figure (6.23) illustrates the process model of ip_icmp and their states' transitions in OPNET. ip_icmp represents the process model of DS_ps process model.

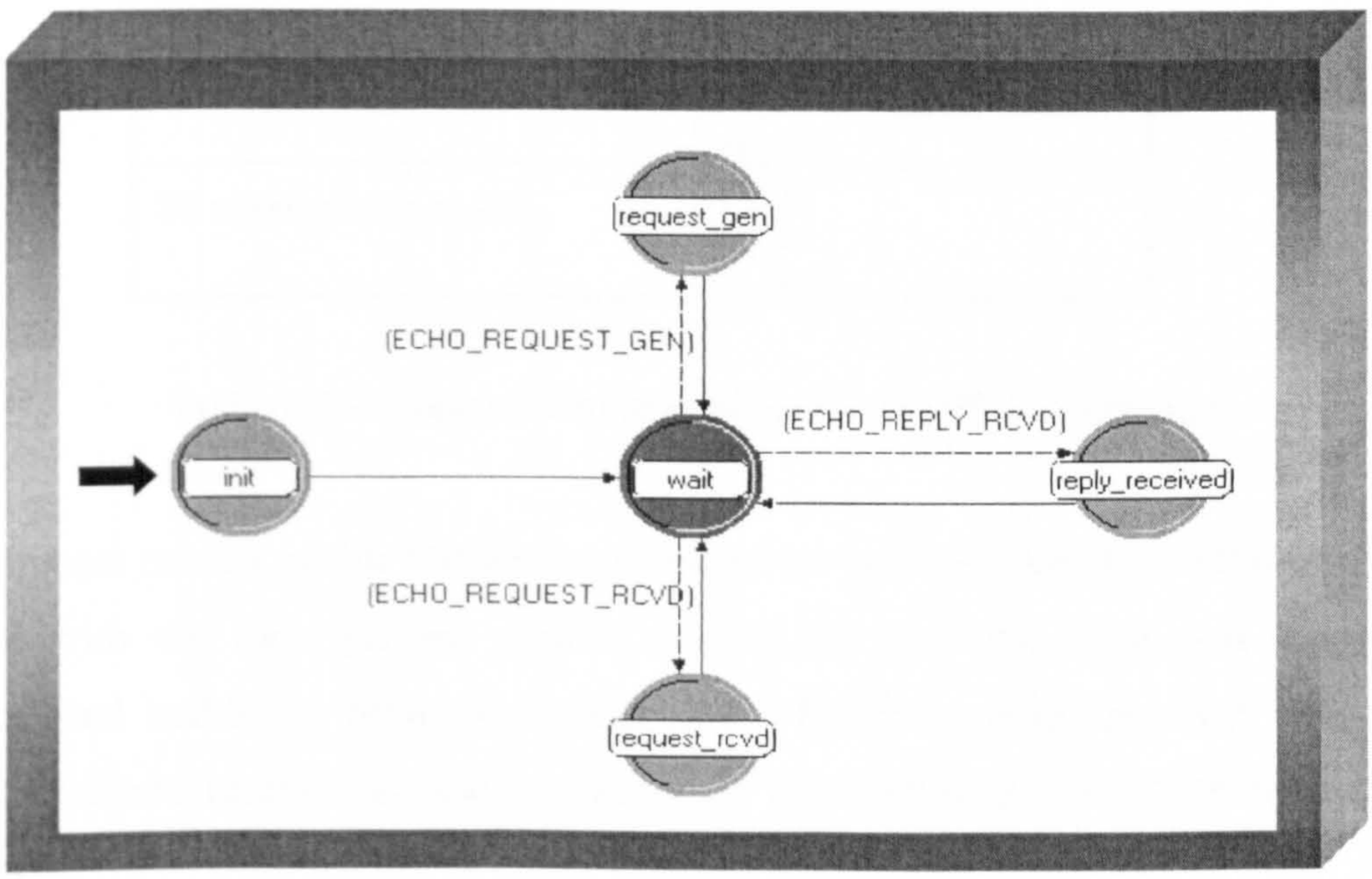


Figure (6.23): ip_icmp process model

6.4 SUMMARY

This chapter has proposed and defined the process models used to implement the ECTP transport protocol and the two proposed models (model (I) and (II)) in the optimised network engineering tool (OPNET) simulator. This provides a comprehensive development environment for modelling and performance of communications networks.

The implementations have followed the OPNET modelling methodology to define each process model. Table (6.25) summarizes the implemented process models in OPNET simulator.

IMPLEMENTED PROCESS MODEL	MODELS
ECTP transport protocol	ectp_manager ectp_connection ectp_QoS_management rsvp aggrsvp
DS router process model	DS_ip DS_ps ip_icmp

Table (6.25): Implemented process models in OPNET simulator

These process models enable OPNET to simulate the performance of ECTP transport protocol with the two mapped models (model (I) and (II)). The next chapter discusses and builds the network models to use ECTP transport protocol with or without resource reservation and compares its performances with UDP transport protocol.

CHAPTER 7

CASE STUDIES AND ANALYSES

7. OVERVIEW

The main objectives of this thesis to enable the ECTP transport protocol with QoS management to support end-to-end QoS for multicast application requirements.

Chapter 5 proposed and defined two models (model (I) and (II)) to enable ECTP transport protocol with end-to-end QoS management using the standard and aggregated RSVP signaling protocol. Also Chapter 6 implemented the proposed models in OPNET simulation tool.

This chapter discusses three case studies to simulate and analyze the behavior and performance of the proposed models over a network using the OPNET simulation tool. The initial step towards the investigation of these case studies is to analyze the behavior of voice application using either the ECTP or the UDP transport protocols with or without resources reservation. This thesis follows three steps to built each case study, which are described as follow:

1. CHOOSING MODELING METHODOLOGY

The method for building the network model varies depending on what information is to be obtained from the performance analysis. There are three methods, which can be described as follows [OPNET_Topology]:

- SINGLE PATH METHOD- only the infrastructure supporting traffic between two devices of interest is represented (e.g. analyses of performance of particular applications using particular traffic parameters).

- PARTIAL PATH METHOD- represents portions of the network in detail while abstracting other sections (e.g. utilization of a backbone network).

- COMPLETE PATH METHOD- is necessary when a particular problem is scaled across the network and it is important to identify the impact of the problem on all devices involved (e.g. deployment of new application).

This research has chosen the single path method because the goal is to investigate the behavior of voice application for particular traffic parameters using either ECTP or UDP transport protocols with and without resource reservation. Three case studies are used to provide a combination of two single path methods over each network model. This work also assumes simplex connections over the networks. Therefore it is necessary to have two receivers or more (in this case only two receivers) instead of one receiver in each single path.

For all case studies: the network links between all routers are set to PPP DS0 (point-to-point Digital signal 0).The network links between senders to the router are set to PPP DS0. The network links between receivers to the switch are set to 10BaseT.

Case studies (I) and (II) both use the Cisco 7204 as a router, because it consists of all the components that are needed for these case studies. This router also has the advantage that its components can be modified in order to handle RSVP aggregation following the DiffServ principle (Chapter 6, section 6.3). Case study (III) assumes the use of the Cisco 7204 for its edge routers and a modified Cisco 7204 router for its border and core routers.

2. CHOOSING TRAFFIC SPECIFICATION

Characterization of network traffic plays a crucial role in the simulation. Traffic information can potentially affect the results and conclusion drawn from the model. There exist two fundamental representations of traffic, which are described as follows [OPNET_Traffic]:

- EXPLICIT TRAFFIC- refers to a packet-by-packet data transfer with each transfer modeled as a discrete event e.g. application performance analysis between two nodes. [OPNET_Modeling]
- BACKGROUND TRAFFIC- refers to analytically modeled traffic that impacts performance of explicit traffic by inducing additional delays e.g. the effect of server relocation on link utilization.

This research has assumed the use of OPNET explicit traffic. It provides explicit application traffic which includes a set of models for generating traffic based on

standard applications such as FTP, HTTP and voice. Explicit traffic modeling provides the most accurate results, since it models all protocols effect. However, this also results in longer simulations and higher memory usage .Each three case studies in this research analysis voice application traffic transmitted between sender and two receivers using either ECTP or UDP transport protocols.

3. SIMULATION

Once the network model and traffic specifications are chosen, the final goal is the simulation. The simulation results for these case studies are to investigate voice application behavior using either ECTP or UDP transport protocols with or without resource reservation.

Table (7.1) summarizes the description of each case study with their corresponding scenarios.

CASE STUDY	PROJECT	SCENARIO	DESCRIPTION
I	Without_RSVP	ECTP_UDP	All nodes use either ECTP or UDP transport protocols without using RSVP resource reservation over voice applications.
II	ECTP_RSVP	UDP	All the nodes use either ECTP enabled by RSVP resource reservation or UDP without RSVP resource reservation over voice applications.
		UDP_RSVP	All nodes use ECTP and UDP enabled by RSVP resource reservation over voice applications.
		ECTP	All the nodes use either ECTP enabled by RSVP resource reservation or ECTP without RSVP resource reservation over voice applications.
III	ECTP_RSVP_DiffServ	UDP	All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or UDP without RSVP resource reservation over voice applications.
		UDP_RSVP	All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation (Ref. Chapter 4) or UDP with RSVP resource reservation over voice applications.
		ECTP	All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or ECTP without RSVP resource reservation over voice applications.
		ECTP_RSVP	All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or ECTP with RSVP resource reservation over voice applications.

Table (7.1): Description of three case studies

7.1 CASE STUDY (I)- ECTP WITHOUT RSVP

Case Study (I) investigates the behavior and performance of the ECTP transport protocol enabled without RSVP resource reservation (Chapter 2, section 2.7) over voice application. In this case study, the ECTP transport protocol is compared with UDP transport protocol without RSVP for voice application traffic delays. Figure (7.1) illustrates network model project "*Without_RSVP*", scenario "*ECTP_UDP*" in OPNET simulator. *

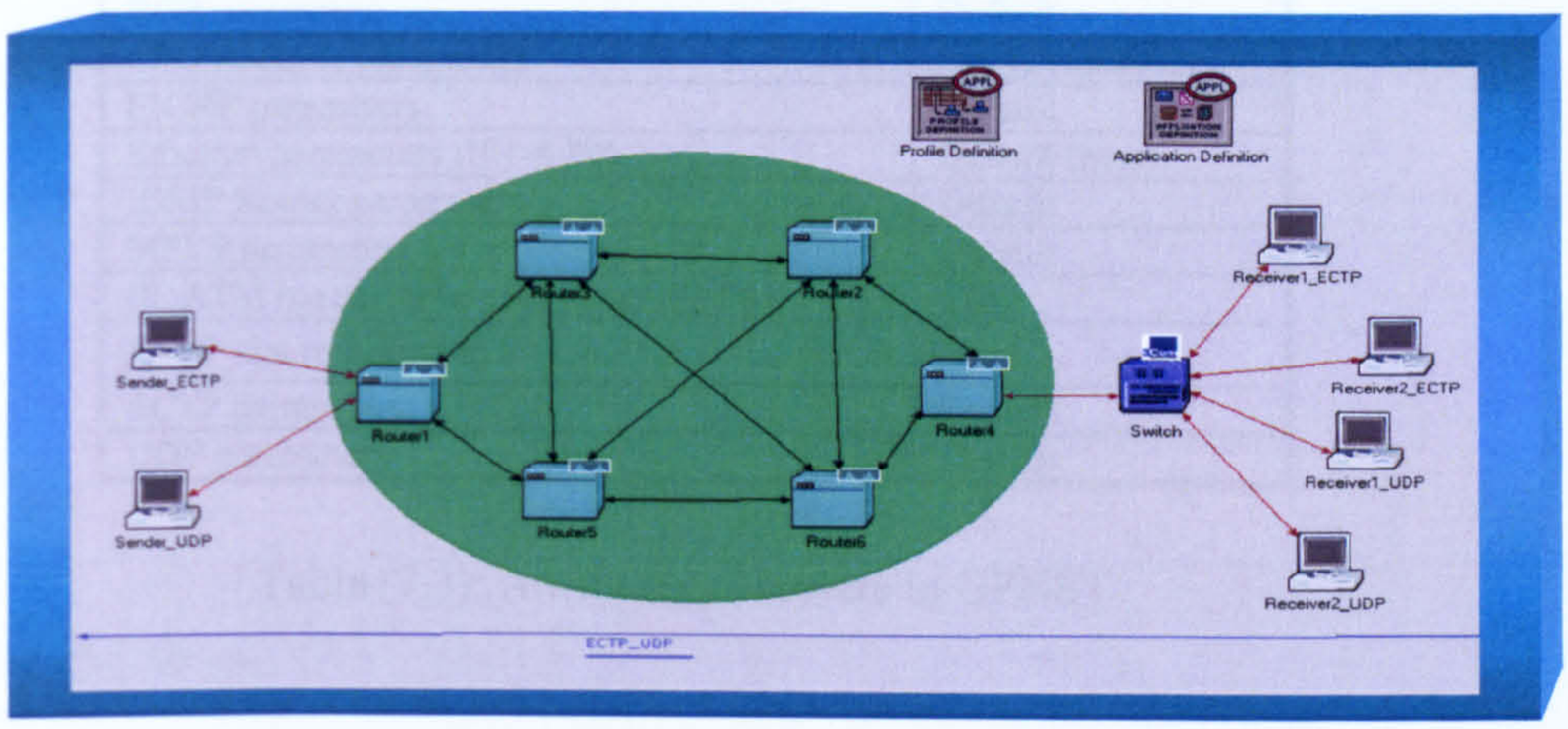


Figure (7.1): OPNET Network model, without_RSVP, Scenario ECTP_UDP

7.1.1 NETWORK MODEL DESCRIPTION

The network model consists of four components, which are described as follows:

1. **WORKSTATIONS** - the network model consists of six workstations, *ppp_wksn_adv*, each of which uses ECTP or UDP as a transport protocol. Table (7.2) summarizes each workstation model and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP Receiver1_ECTP Receiver2_ECTP	ppp_wksn_adv_ECTP	ECTP Control part	ECTP_manager ECTP_connection
Sender_UDP Receiver1_UDP Receviver2_UDP	ppp_wksn_adv_UDP	UDP	rip_udp_v3

Table (7.2): Description of workstation models

* Appendix 6 has shown the baseline case study, which only uses ECTP and UDP without any routers.

2. **ROUTERS-** the network model consists of 6 routers, *CISCO 7204 (CS_7204_4S_a1_e8_f1_Sl8_adv).* All the routers use UDP as a transport protocol in the core network. Table (7.3) shows the corresponding attributes in the routers.

ROUTER ATTRIBUTES	VALUE
ARP parameters (IF1-9 P0)	Default
ATM ALL 3/4 segmentation rate (IF0 P0	Peak cell rate
ATM ALL 5 segmentation rate (IF0 P0)	Peak cell rate
ATM ABR Parameter (IF0 P0)	None
ATM address (IF0 P0)	Auto assigned
ATM CAC address	Default
BGP parameters	Default
CPU resource parameters	Single processor
EIGRP parameters	Default
Ethernet parameters (IF1-8 P0)	Default (host)
IGMP Router parameters	Default
MPLS parameters	Default
IP_ATM inactivity release time (IF0 P0)	120
RSVP slot information	None
ECTP parameters	Default
UDP parameters	Default

Table (7.3): Attributes of routers in OPNET

3. **APPLICATION DEFINITION** - OPNET simulator uses applications for generating traffic patterns at each node, which should be created and configured in the application definitions. Table (7.4) describes the voice application parameters that have been defined in this model.

APPLICATION DEFINITION	DESCRIPTION	PARAMETERS	VALUES
Voice RSVP Not Used	Voice Application	Silence Length (second)	None
		Talk Spurt length (seconds)	None
		Symbolic Destination Name	Voice Destination
		Encoder Scheme	G.723.1
		Voice Frames per Packet	1
		Type of service_ECTP	Standard
		Type of service_UDP	Best effort
		RSVP parameters	None
		Traffic Mix (%)	All Discrete

Table (7.4): Description of application definition

4. **PROFILE DEFINITION** - once the applications have been defined, the next step is to include the application definitions in the profile definition that is later deployed to workstations. In this model, the profile name is "*voice_unreserved*",

which means the application does not support the RSVP reservation. Table (7.5) describes the profile definition and configuration parameters.

PROFILE DEFINITION	PROFILE CONFIGURATION	PARAMETERS	APPLICATION	PARAMETERS
voice_unreserved	Application	Voice RSVP Not used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		

Table (7.5): Description of profile definition and configuration

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.6) describes the supported profiles in the all workstations.

WORKSTATION NAME	APPLICATION ATTRIBUTES
	SUPPORTED PROFILES
Sender_ECTP Sender_UDP	voice_unreserved
Receiver1_ECTP Receiver2_ECTP Receiver1_UDP Receiver2_UDP	none

Table (7.6): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** OPNET simulator creates explicit application traffic which includes a set of models for generating traffic. This traffic has been specified inside the voice application definition (Table (7.4)). Type of service assigned to the packets sent from the both senders (Sender_ECTP and Sender_UDP). It represents a session attribute which allows packets to be processed faster in ip queues (it is an integer between 0-252, 252 being the highest priority).

Traffic between *Sender_ECTP*, *Receiver1_ECTP* and *Receiver2_ECTP* uses Standard type of service with the integer priority value of 64 and delay

parameter. Traffic between *Sender_UDP*, *Receiver1_UDP* and *Receiver2_UDP* uses Best effort type of service with the integer priority value of 0 and delay parameter.

7.1.2 COLLECTING STATISTICS

Once all the components have been defined in this scenario, the next step is to collect available statistics on the network model. OPNET simulator supports predefined application statistics on the network model, which are described as follows:

- **Global Statistics** collects information about how an application is performing in general across all nodes.
- **Node Statistics** collects information about how each individual node is performing.
- **Link Statistics** collects information under the "point-to-point" category, which determines if any parts of network are congested and thus causing additional delays in the application's performance.

Table (7.7) defines the collected statistics in this network model.

STATISTICS	COLLECTION	PARAMETERS
Global Statistics	None	
Node Statistics	Voice application	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice called party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice calling party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)

Table (7.7): Collected statistics in network model

7.1.3 SIMULATION SETUP

Before running the simulation, OPNET simulator will setup values for simulation attributes. Simulation duration is set to 1.0 hour. Table (7.8) describes the simulation setup attributes for this network model.

SIMULATION SETUP ATTRIBUTES	VALUE
APR Sim Efficiency	Enabled
BGP Sim Efficiency mode	Enabled
IGM Sim Efficiency	Enabled
IP Dynamic Routing Protocol	Default
IP Routing Table Export/Import	Not Used
OSPF Sim Efficiency	Enabled
RIP Sim Efficiency	Enabled
RSVP Sim Efficiency	Disabled

Table (7.8): Attributes of simulation setup

7.1.4 SIMULATION RESULTS

Figure (7.2) illustrates the simulation results of end-to-end traffic delay with the voice application traffic in the ECTP_UDP scenario. The ECTP (pink) graph represents the variations of traffic delay on the network using ECTP transport protocol without RSVP. The UDP (blue) graph represents the variation of traffic delay parameter on the network using UDP transport protocol without RSVP.

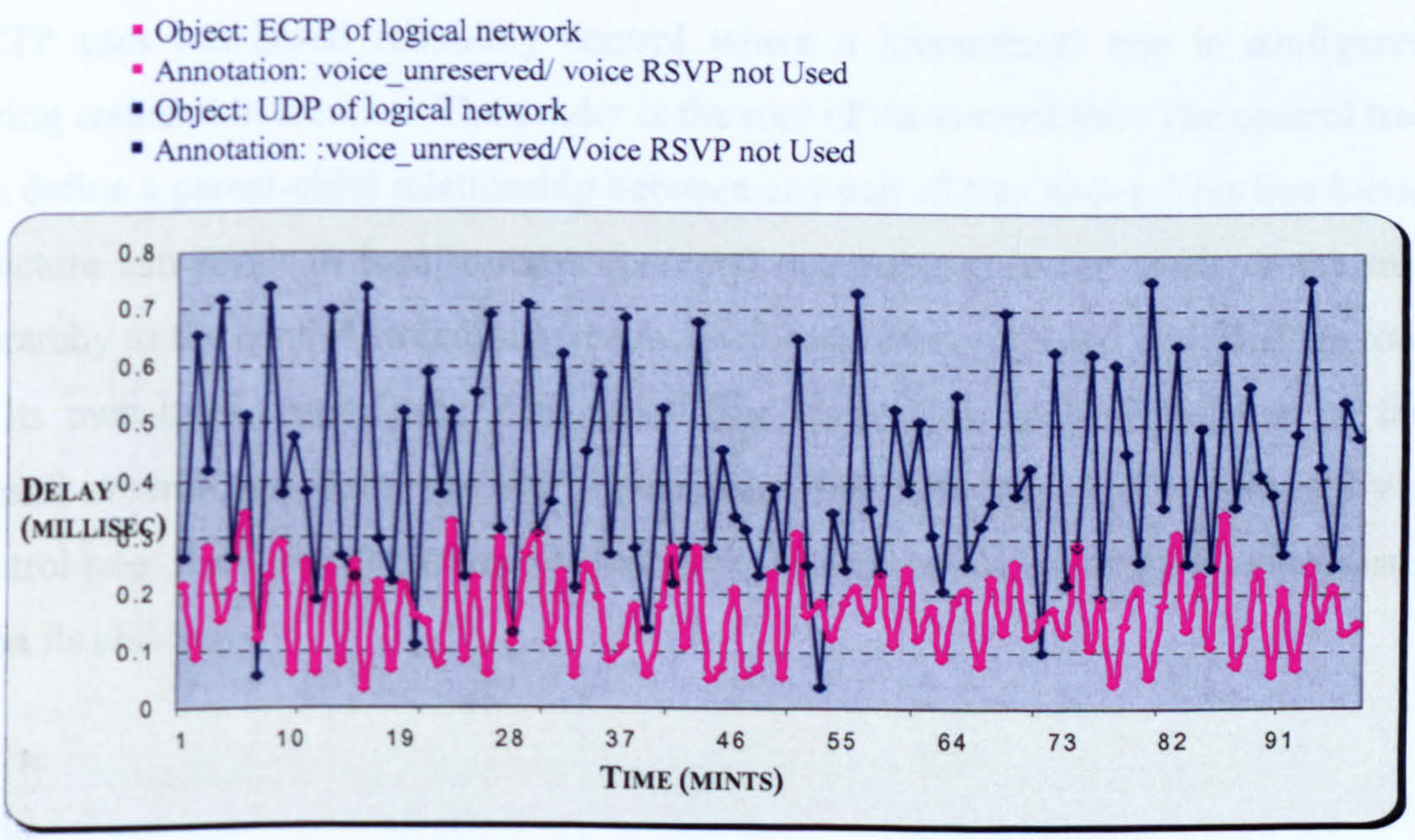


Figure (7.2): Voice application end-to-end traffic Delay (millsec)

7.1.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.2) compares the voice application's traffic delay experienced using either ECTP or UDP transport protocols without RSVP resource reservation. As it mentioned in Chapter 2, section (2.4.3), all multicast applications are delay-sensitive because the transmitted information needs to be re-played at the receiver end in real-time. Therefore, it is important to measure the traffic delay on all networks.

As shown in the ECTP graph (pink) during the 91 minutes simulation the traffic delay variation remained between *0.03254* (*millsec*) minimums and *0.34082* (*millsec*) maximums. However, by looking at the UDP graph (blue) the traffic delay variation fluctuates between *0.056* (*millsec*) minimums and *0.751 (millsec*) maximums. The overall simulation results show that the ECTP transport protocol performs better than UDP transport protocol to deliver voice application traffic over the network. As mentioned in Chapter 2, section 2.7. , the main reason is that because of the ECTP transport protocol has tree-based functionality; it could provide reliability and scalability (over nodes) for voice applications compare to UDP transport protocol, hence, the traffic delays during the transmission of voice applications are reduced by using ECTP transport protocol.

ECTP uses tree-based reliability control where a hierarchical tree is configured during connection creation. The sender is the root of the control tree. The control tree can define a parent-child relationship between any pair of tree nodes. This tree-based structure can result in local owners (parents) occurring at lower levels in the tree hierarchy as the control structure extends. Each local owner created becomes the root of its own local control tree. The connection owner will then be the root of the overall control tree. Error control is performed for each local group defined by a control tree. Each parent retransmits lost data, in response to retransmission requests from its children.

7.2 CASE STUDY (II)- ECTP ENABLED BY STANDARD RSVP

Case Study (II) investigates the behavior and performance of ECTP transport protocol enabled by the standard RSVP resource reservation (Chapter 5, section 5.4) using voice application. In this case study, the ECTP transport protocol enabled by standard RSVP is compared with either the ECTP or UDP transport protocol (with or without RSVP) over traffic delays for voice application. Case study (II)-project *"ECTP_RSVP"* consists of three scenarios, which are as follow:

- **SCENARIO (1)- UDP:** All the nodes use either ECTP enabled by RSVP resource reservation or UDP without RSVP resource reservation for voice applications.
- **SCENARIO (2)- UDP_RSVP:** All nodes use either ECTP or UDP enabled by RSVP resource reservation for voice applications.
- **SCENARIO (3)- ECTP:** All the nodes use either ECTP enabled by RSVP resource reservation or ECTP without RSVP resource reservation for voice applications.

7.2.1 SCENARIO (1)- UDP

In this scenario, all the nodes use either ECTP enabled by RSVP resource reservation (Chapter 5. section 5.4) or UDP without RSVP resource reservation. Figure (7.3) illustrates network model project *"ECTP_RSVP"*, scenario (1) *"UDP"*.

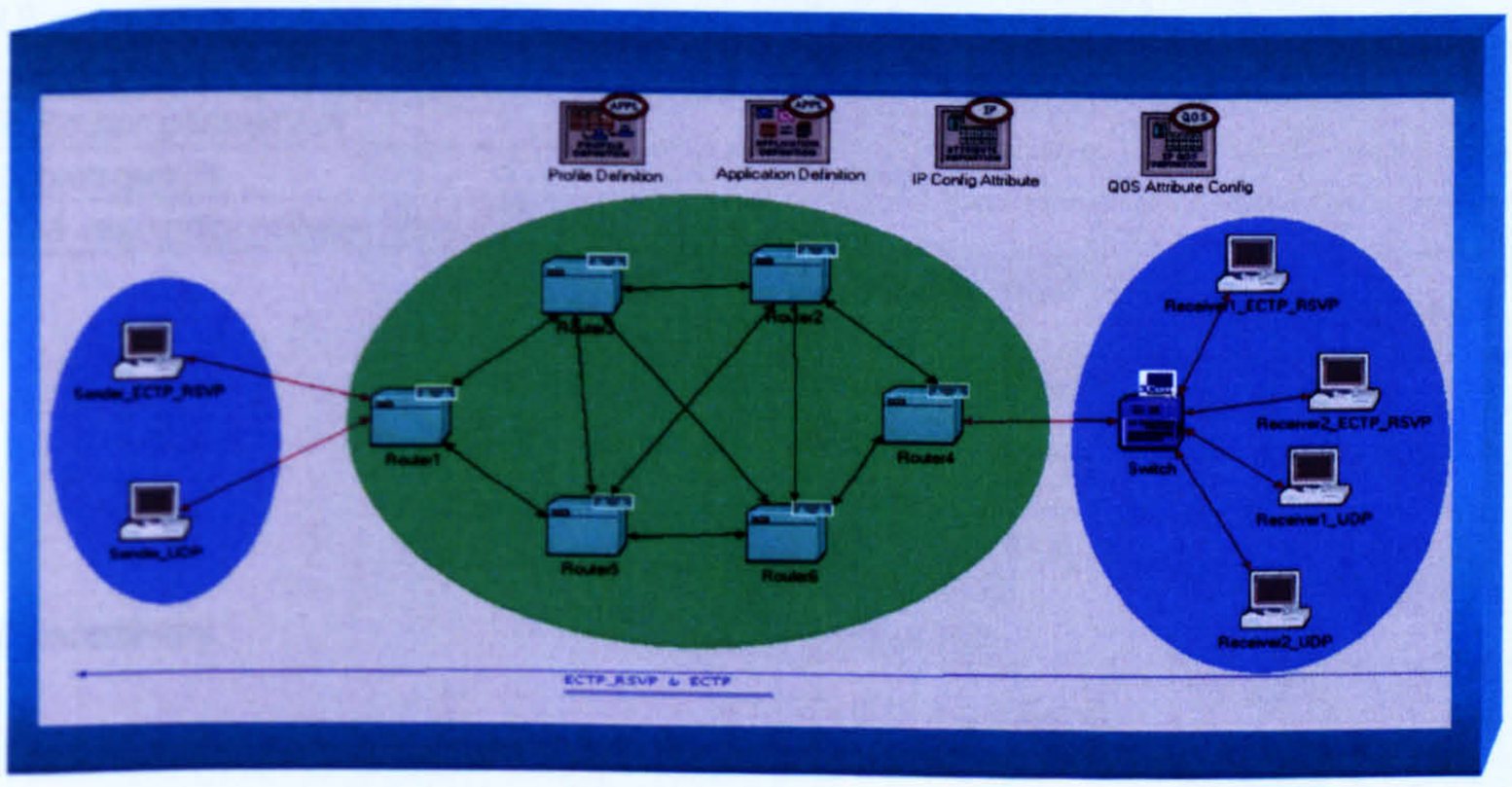


Figure (7.3): OPNET Network model project ECTP_RSVP, Scenario (1)-UDP

7.2.1.1 NETWORK MODEL DESCRIPTION

The network model consists of six components, which are described as follows:

1. **WORKSTATIONS-** the network model consists of six workstations; *ppp_wksn_adv*. Table (7.9) summarizes each workstation model and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL	PROCESS MODEL
Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	ppp_wksn_adv_ECTP_RSVP	ECTP QoS management (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management rsvp
Sender_UDP Receiver1_UDP Receiver2_UDP	ppp_wksn_adv_UDP	UDP	rip_udp_v3

Table (7.9): Description of workstation models

2. **ROUTERS-** the network model consists of 6 routers, *CISCO 7204 (CS_7204_4S_al_e8_fl_Sl8_adv)*. All routers have been chosen randomly to use either ECTP with RSVP or UDP without RSVP in the core network. Table (7.10) shows the corresponding attributes in the routers.

ROUTER ATTRIBUTES	VALUE		
ATM ALL 3/4 segmentation rate (IF0 P0	Peak cell rate		
ATM ALL 5 segmentation rate (IF0 P0)	Peak cell rate		
ATM address (IF0 P0)	Auto assigned		
BGP parameters	Default		
CPU resource parameters	Single processor		
EIGRP parameters	Default		
Ethernet parameters (IF1-8 P0)	Default (host)		
IGMP Router parameters	Default		
MPLS parameters	Default		
IP_ATM inactivity release time (IF0 P0)	120		
ECTP parameters	Maximum segment size	Auto assigned	
	Receiver buffer (bytes)	8760	
	Receive buffer usage threshold	0.0	
	Delay ACK mechanism	Segment/clock based	
	RSVP parameters	Waiting time (sec)	1.0
		Refresh internal (sec)	30
		Lifetime multiplier	10
UDP parameters	Default		

Table (7.10): Attributes of routers in OPNET

3. APPLICATION DEFINITION- table (7.11) describes the application definition parameters in scenario (1)_UDP.

APPLICATION DEFINITION	DESCRIPTION	PARAMETERS	VALUES	
Voice RSVP Not Used	Voice Application	Silence Length (second)	default	
		Talk Spurt length (seconds)	default	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Best Effort	
		RSVP parameters	None	
		Traffic Mix (%)	All Discrete	
Voice RSVP Used	Voice Application	Silence Length (second)	default	
		Talk Spurt length (seconds)	default	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Reserved	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	

Table (7.11): Description of application definition

4. PROFILE DEFINITION- table (7.12) describes the profile definition of scenario (1)_UDP.

PROFILE DEFINITION	PROFILE CONFIGURATION	PARAMETERS	APPLICATION	PARAMETERS
voice_unreserved	Application	Voice RSVP Not used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		
voice_reserved	Application	Voice RSVP used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		

Table (7.12): Description of profile definition and configuration

Once the applications and profiles have been configured, it is necessary to deploy to the individual workstations. Table (7.13) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES			
	RSVP PARAMETERS			SUPPORTED PROFILES
Sender_ECTP_RSVP	Voice application	RSVP Status	Enabled	voice_reserved
		Profile list	Default	
Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	Voice application	RSVP Status	Enabled	None
		Profile list	Default	
Sender_UDP	none			voice_unreserved
Receiver1_UDP Receiver2_UDP	none			None

Table (7.13): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.11). Traffic between *Sender_ECTP_RSVP*, *Receiver1_ECTP_RSVP* and *Receiver2_ECTP_RSVP* uses Reserved type of service with the integer priority value of 224 and delay parameter. The reservation has been made for traffic in both directions. The traffic generated by *Sender_ECTP_RSVP* is described as having a bandwidth of 5,000(bytes/sec) and a buffer size of 10,000 bytes (Table 7.16). Traffic between *Sender_UDP*, *Receiver1_UDP* and *Receiver2_UDP* uses Best effort type of service with the integer priority value of 0 and delay parameter.

6. **IP CONFIGURATION-** table (7.14) defines the IP configuration attributes in scenario (1)_UDP.

IP CONFIGURATION ATTRIBUTE	VALUE
IP compression Information	Standard compression
IP multicast RP Information	Default
IP Ping parameters	Default
IP Route table export	Once at end of simulation

Table (7.14): Description of IP configuration attributes

7. **RSVP CONFIGURATIONS-** attributes describing RSVP parameters set in two objects:

- **QOS ATTRIBUTE CONFIGURATION OBJECT-** There is two sets of attributes defined in this object. Table (7.15) defines the QoS attributes configuration object.

QOS ATTRIBUTE CONFIGURATION	VALUES
CAR Profiles	Defaults
Custom Queuing Profiles	Defaults (Standard Schemes)
FIFO Profiles	Defaults (Standard Schemes)
Priority Queuing Profiles	Defaults (Standard Schemes)
RSVP Flow Specification	**See Table (7.16)**
RSVP Profiles	**See Table(7.17)**
WFQ Profiles	Defaults (Standard Schemes)

Table (7.15): Description of QoS configuration objects

- **RSVP FLOW SPECIFICATION-** RSVP data flow is assigned to the *Outbound_Flow* attributes and *Inbound_Flow* attributes which they define, the application's bandwidth and buffer size for reservations respectively. Bandwidth (bytes/sec) specifies the amount of traffic generated by the application at the IP level. This value is set to be token rate in flow specification of Path and Resv messages. When the reservation is made using this flow specification, this value is set as the reserved bandwidth for the session.

  The amount of data that should be buffered can be configured using the buffer size (bytes) attributes. This value is used as bucket size and is set in Path or Resv messages for this session. The amount of data traffic sent over all time periods should not exceed $r^*.T+b$ where r is the token bucket rate which is the value of bandwidth attribute, b is token bucket size which is the value of buffer size, and T is the measurement interval in seconds (Chapter 3, section 3.4.2). Table (7.16) defines the RSVP flow specification parameters in OPNET simulator.

RSVP FLOW SPECIFICATION NAME	BANDWIDTH (BYTES/SEC)	BUFFER SIZE (BYTES)
Outbound_Flow	5,000	5,000
Inbound_Flow	10,000	50,000
Reservation	5,000	10,000

Table (7.16): RSVP flow specification

- **RSVP PROFILES-** The threshold (byte/sec) attribute is used when a host application only requests a reservation if a Path message indicates that the traffic

load is higher than the value specified in this attribute. The reservation style attribute includes setting for *fixed-wildcard*, *shared_explicit* and *fixed_filter* reservation styles (Chapter 3, section 3.5.3). The reservation parameters are assigned a specific data flow and sender list of profiles.

The RSVP model uses admission-control-based functions to process reservation requests. If the network is unable to grant a reservation request, the RSVP profile's Retry Policy compound attribute will determine how the receiving host responds. If the retry attribute is set to "Do Not Retry", the application uses a best effort QoS. Otherwise, the request is resent according to the Retry timer (sec) and reservation parameters. Table (7.17) defines the RSVP profiles parameters.

RSVP PROFILE NAME	THRESHOLD (BYES/SEC)	RESERVATION STYLE	RESERVATION PARAMETERS	RETRY POLICY
Wild_card_profile	4,000	Wild card	Specified by sender	Do Not Retry
Shared_explicit_profile	None	Shared explicit	see table (7.18)	see table(7.19)
Fixed_filter_profile	10,000	Fixed filter	Specified by sender	Do Not Retry

Table (7.17): RSVP profile parameters

Table (7.18) defines shared_explicit_profile reservation parameters.

Shared_explicit_profile RESERVATION PARAMETERS	VALUES
Flow description	specified by sender
Sender list	Sender_ECTP

Table (7.18): Reservation parameters

Table (7.19) defines shared_explicit_profile retry policy parameters.

Shared_explicit_profile RETRY POLICY	VALUES	
Retry	Retry	
Retry timer (sec)	30	
Reservation Parameters	Flow description	Reservation
	Sender list	Any sender

Table (7.19): Retry policy parameters

- **RSVP TRAFFIC CONTROL PROCESS-** OPNET's RSVP model supports Controlled load service (Chapter 3, section 3.4.2). This service is supported for Weighted Fair Queuing (WFQ) and Priority Queuing schemes (Chapter 3, section 3.7). In the OPNET implementation, WFQ and Priority Queuing create or delete

a queue with requested parameters (bandwidth and buffer size) when needed. Therefore, there is one queue per reservation and the number of queues is dynamic and depends on the number of reservations in place on a node.

- **APPLICATION ATTRIBUTE CONFIGURATION OBJECT-** RSVP parameters for Path message is defined in this object. The RSVP status default is none, whereas in this model RSVP status sets as enabled. Table (7.20) shows RSVP data flows in the application attribute configuration object.

RSVP PARAMETERS	VALUE
RSVP Status	Enabled
Outbound Flow	Outbound_Flow
Inbound Flow	Inbound_Flow

Table (7.20): RSVP data flows

- **ENABLING RSVP ON WORKSTATIONS-** Once the RSVP is configured on the applications and profile configuration objects it is ready to be deployed on individual workstation. Table (7.21) defines the supported profiles in the all workstations.

WORKSTATION ATTRIBUTES	SUPPORTED PROFILES
Sender_ECTP Sender_UDP	voice_unreserved
Receiver1_ECTP Receiver2_ECTP Receiver1_UDP Receiver2_UDP	none

Table (7.21): Supported Profiles

7.2.1.2 COLLECTING STATISTICS

As was discussed in section (7.1.2), OPNET supports predefined application statistics on the network model. Once all the RSVP configuration objects have been defined, the next step is to collect the RSVP statistics. The following statistics related to the RSVP process are defined as follows:

- **RSVP global statistics** capture the total amount of RSVP traffic sent and received in the whole network. These statistics show the message overhead of RSVP processing in the network.

- **RSVP node statistic** can be divided into three groups:
  1. Message statistics show the number of RSVP messages (Path, Resv, Confirmation and total) received or sent by a node.
  2. State statistics show the number of RSVP states (Path, Resv and Blockade), collected as average value over a period of time.
  3. Reservation statistics show the number of successful or rejected reservations, collected as an average values over a period of time.

- **IP interface RSVP node statistics** show the RSVP allocated bandwidth (bytes/sec) and buffer size (bytes) for each interface.

Table (7.22) defines the collected statistics in scenario (1)-UDP.

STATISTICS	COLLECTION	PARAMETERS
Global Statistics	None	None
Node Statistics	RSVP	Path messages received (packet/sec)
		Path messages sent(packet/sec)
		Resv Conf messages received (packet/sec)
		Resv Conf messages sent(packet/sec)
		Resv messages received (packet/sec)
		Resv messages sent (packet/sec)
	IP Interface	RSVP allocated bandwidth (bytes/sec)
	Voice application	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice called party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice calling party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)

Table (7.22): Collected statistics in network model

7.2.1.3 SIMULATION SETUP

Before running the simulation, OPNET will set up values for the simulation attributes. Simulation duration is set to 1.0 hour. Table (7.23) describes the simulation setup attributes for this network model.

SIMULATION SETUP ATTRIBUTES	VALUE
APR Sim Efficiency	Enabled
BGP Sim Efficiency mode	Enabled
IGM Sim Efficiency	Enabled
IP Dynamic Routing Protocol	Default
IP Routing Table Export/Import	Not Used
OSPF Sim Efficiency	Enabled
RIP Sim Efficiency	Enabled
RSVP Sim Efficiency	Enabled

Table (7.23): Attributes of simulation setup

7.2.1.4 SIMULATION RESULTS

Figure (7.4) illustrates the simulation results of end-to-end traffic delay over voice application in the UDP scenario. The ECTP_RSVP (green) graph represents the variations of traffic delay parameter on the network using ECTP transport protocol with RSVP. The UDP (blue) graph represents the variation of traffic delay parameter on the network using UDP transport protocol without RSVP.

- Object: ECTP_RSVP of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object: UDP of logical network
- Annotation: voice_unreserved/Voice RSVP not Used

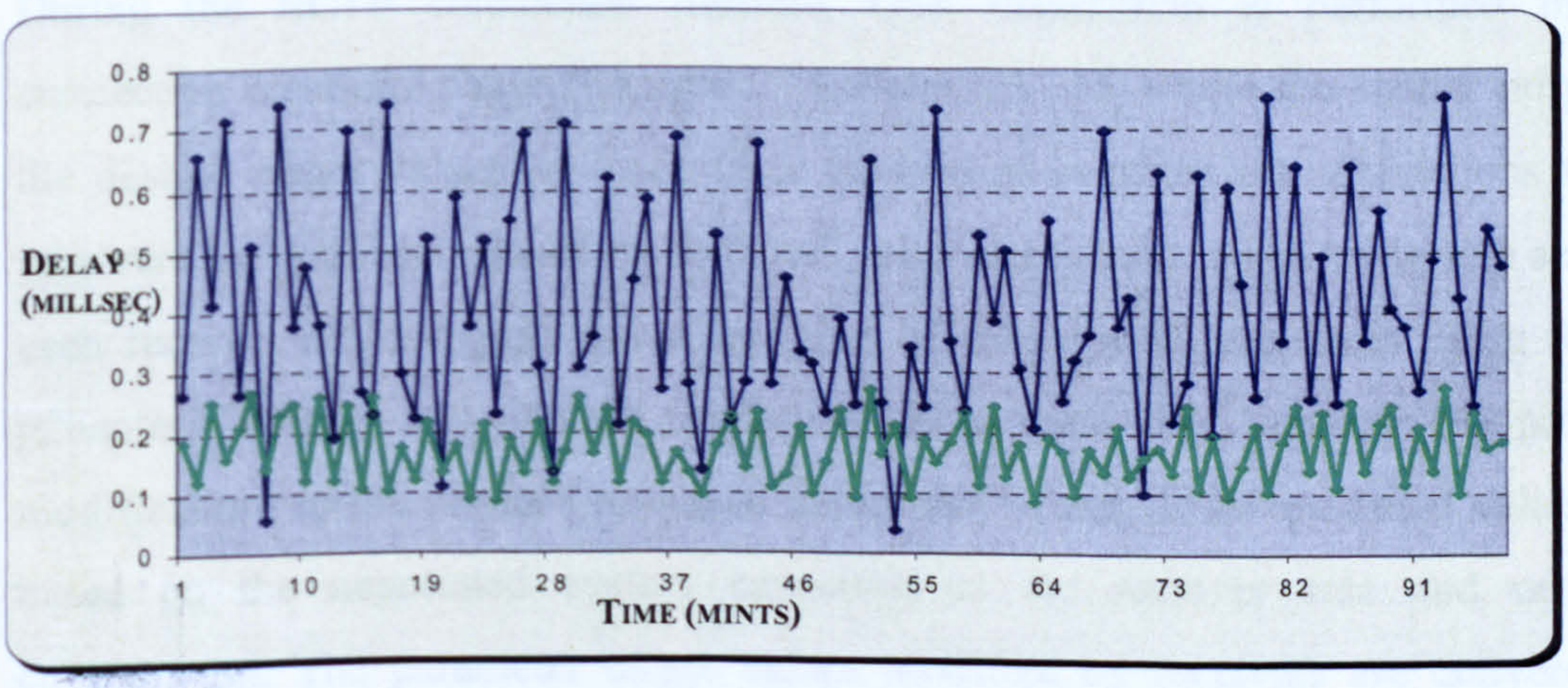


Figure (7.4): Voice application. Packet End-to-End Delay (millsec)

7.2.1.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.4) compares the voice application's traffic delay experienced using ECTP transport protocol with RSVP resource reservation and UDP transport protocol without RSVP resource reservation. As mentioned in Chapter 3, section (3.5), RSVP protocol is a signaling mechanism, which typically carries QoS requests from one end of network to the other along the same path traversed by the data that requires the QoS resources. Also, chapter 5, section (5.4) has proposed enabling ECTP with QoS management using RSVP signaling protocol.

As shown on the ECTP graph (pink) during the 91 minutes simulation the traffic delay variation remained steady between *0.08419* (*millsec*) minimums and *0.2730* (*millsec*) maximum. However, by looking at the UDP graph (blue) the traffic delay variation fluctuates between *0.056* (*millsec*) minimums and *0.751* (*millsec*) maximum. The overall simulation results show that the ECTP transport protocol with the RSVP resource reservation performs better than UDP transport protocol without RSVP resource reservation to deliver voice application over the network. The main reason is that the ECTP tree-based architecture (Chapter 2, section 2.7), and QoS management functionality (Chapter 4, section 4.1), i.e. reserving resources (RSVP), could provide better reliability and scalability (over nodes) along with dynamic admission control for voice applications compared to the UDP transport protocol. Hence, the traffic delays during the transmission of voice applications are reduce by ECTP transport protocol enabled by standard RSVP.

During the ECTP connection lifetime, QoS negotiation is performed in the connection creations phase (Chapter 2, section 2.7.2.1), where the sender proposes the desired target values for each QoS parameters required by applications to all receivers by multicast. Based on the QoS parameter values proposed by the sender, each receiver might trigger the reservation of the network resources using RSVP (Chapter3, section 3.5). If QoS negotiation is enabled, each receiver can propose modifications to the sender's proposed parameter values. These modified values are based on the negotiated system capacities at the receiver side and network environment. The parameter target values modified by receivers are delivered to sender via ACK messages.

7.2.2 SCENARIO (2) -UDP_RSVP

In this scenario, all nodes use either ECTP or UDP enabled by RSVP resource reservation. Figure (7.5) illustrates network model project "*ECTP_RSVP*", scenario (2) "*UDP_RSVP*".

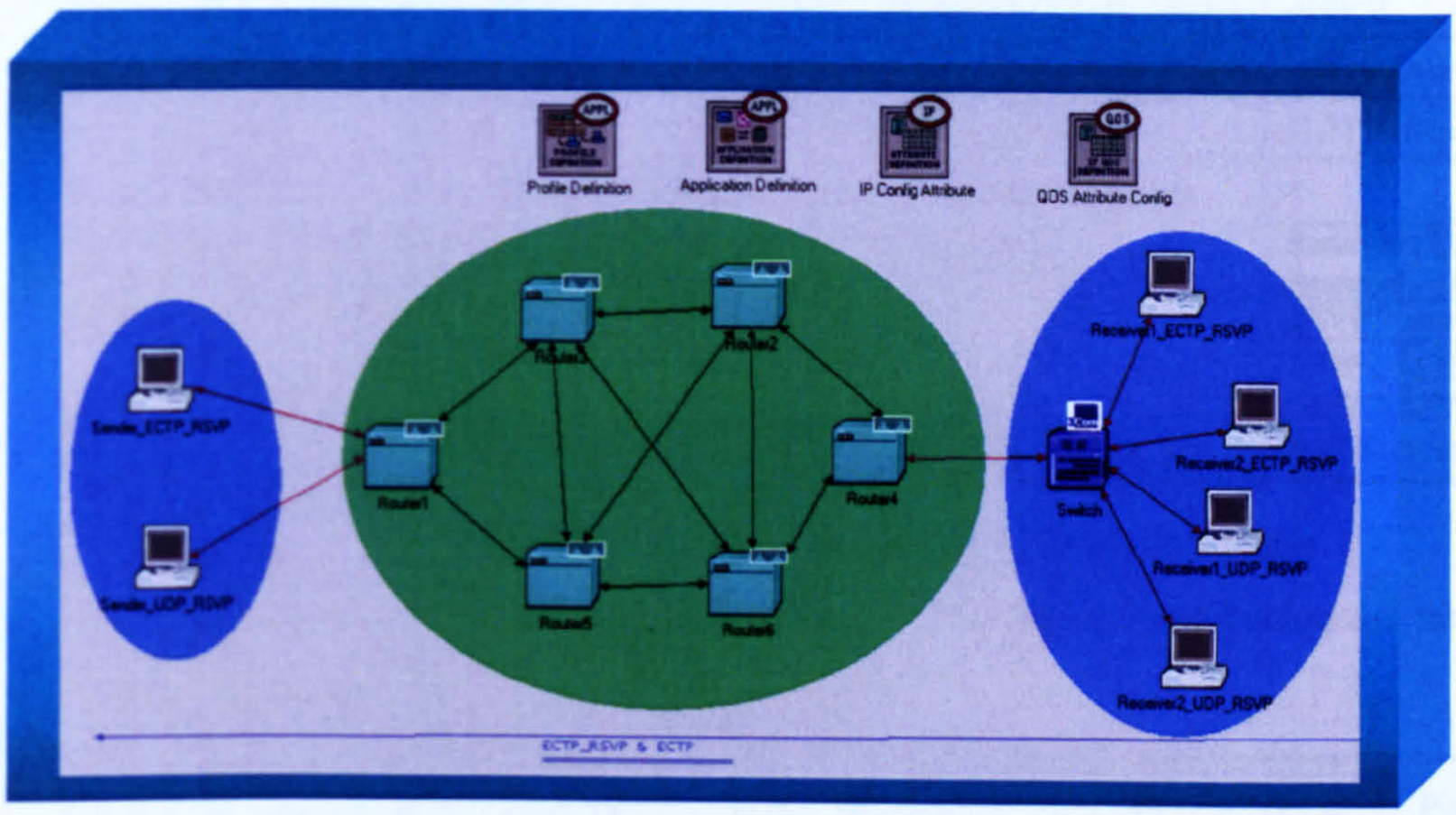


Figure (7.5): OPNET Network model project ECTP_RSVP, Scenario (1)-UDP_RDVP

7.2.2.1 NETWORK MODEL DESCRIPTION

The network model consists of six components, which are described as follows:

1. **WORKSTATIONS**-the network model consists of six workstations; *ppp_wksn_adv*. Table (7.24) describes each workstation model and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	ppp_wksn_adv_ECTP_RSVP	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management rsvp
Sender_UDP_RSVP Receiver1_UDP_RSVP Receiver2_UDP_RSVP	ppp_wksn_adv_UDP_RSVP	UDP	rip_udp_v3 rsvp

Table (7.24): Description of workstation models

2. **ROUTERS**- See section (7.2.1.1), Router.
3. **APPLICATION DEFINITION**- Table (7.25) describes the application definition parameters in scenario (2), UDP_RSVP.

APPLICATION DEFINITION	DESCRIPTION	PARAMETERS	VALUES	
Voice RSVP Used	Voice Application	Silence Length (second)	default	
		Talk Spurt length (seconds)	default	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Reserved (224) Reserved (208)	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	

Table (7.25): Description of application definition

4. **PROFILE DEFINITION**- Table (7.26) describes the profile definition of scenario (2), UDP_RSVP.

PROFILE DEFINITION	PROFILE CONFIGURATION	PARAMETERS	APPLICATION	PARAMETERS
voice_reserved	Application	Voice RSVP used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		

Table (7.26): Description of profile definition and configuration

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.27) describes the application attributes on each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES			
	RSVP PARAMETERS			SUPPORTED PROFILES
Sender_ECTP_RSVP Sender_UDP_RSVP	Voice application	RSVP Status	Enabled	voice_reserved
		Profile list	Default	
Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP Receiver1_UDP_RSVP Receiver2_UDP_RSVP	Voice application	RSVP Status	Enabled	none
		Profile list	Default	

Table (7.27): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.25). Traffic between *Sender_ECTP_RSVP*, *Receiver1_ECTP_RSVP* and *Receiver2_ECTP_RSVP* uses Reserved type of service with the integer priority value of 224 and delay parameter. The reservation has been made for traffic in both directions. The Traffic between *Sender_UDP_RSVP*, *Receiver1_UDP_RSVP* and *Receiver2_UDP_RSVP* uses Reserved type of service with the integer priority value of 208 and delay parameter.
6. **IP CONFIGURATION-** See section (7.2.1.1), IP configuration.
7. **RSVP CONFIGURATION-** See section (7.2.1.1), RSVP configuration.

7.2.2.2 COLLECTING STATISTICS - See section (7.2.1.2), collecting statistics.

7.2.2.3 SIMULATION SETUP- See section (7.2.1.3), simulation setup

7.2.2.4 SIMULATION RESULTS

Figure (7.5) illustrates the simulation results of end-to-end traffic delay over voice application in UDP_RSVP scenario. The ECTP_RSVP (green) graph represents the variations of the traffic delay parameter on the network using ECTP transport protocol with RSVP. The UDP (red) graph represents the variation of traffic delay parameter on the network using UDP transport protocol with RSVP.

- object: ECTP_RSVP of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object: UDP_RSVP of logical network
- Annotation: :voice_reserved/Voice RSVP Used

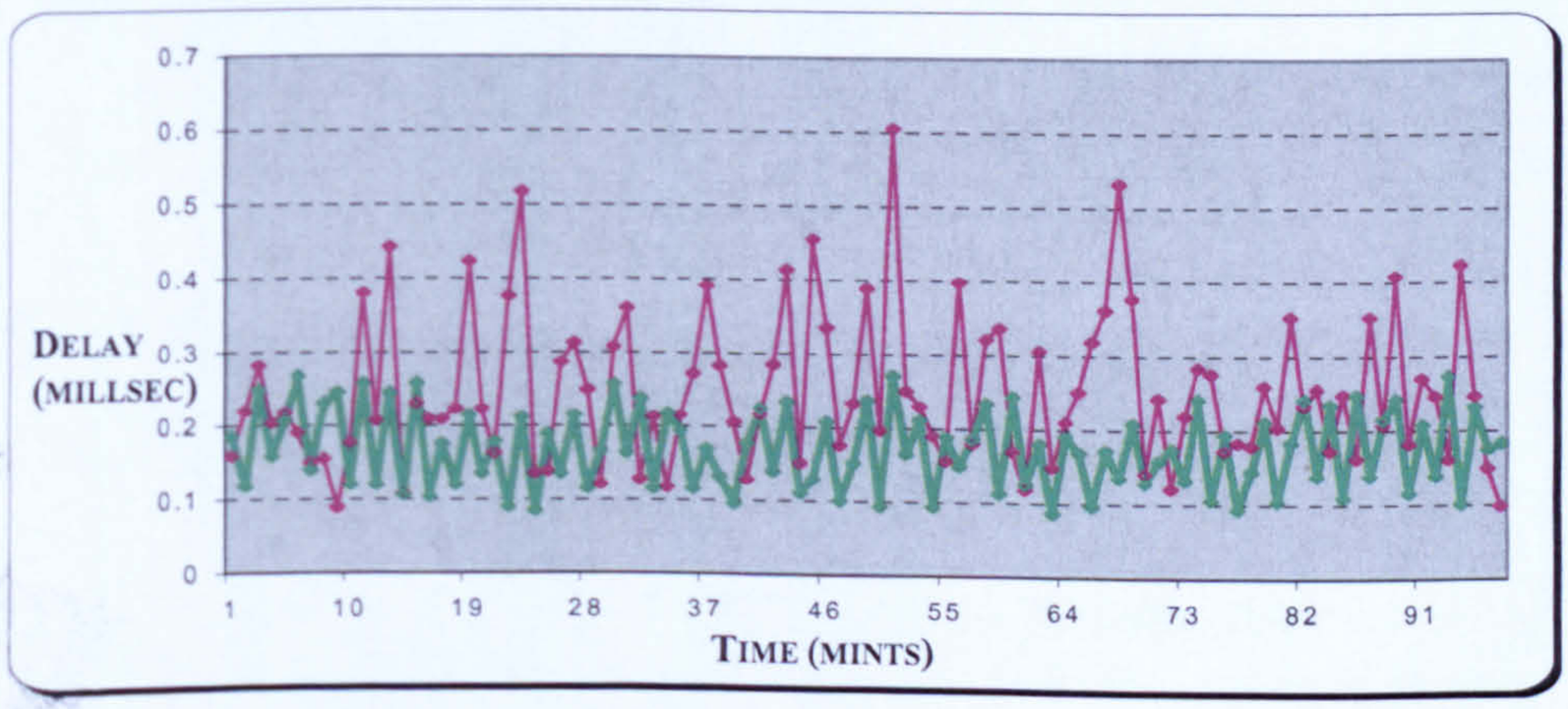


Figure (7.5): Voice application. Packet End-to-End Delay (millsec)

7.2.2.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.5) compares the voice application's traffic delay experienced using either ECTP or UDP transport protocol with RSVP resource reservation. As it shown in the ECTP graph (green) during the 91 seconds simulation the traffic delay variation stayed steady between *0.08419* (sec) minimums and *0.2730* (sec) maximums.

However, by looking at the UDP graph (blue) the traffic delay variation fluctuates between *0.0897* (sec) minimums and *0.6046* (sec) maximums. The overall simulation results show that the ECTP transport protocol with the RSVP resource reservation performs better than UDP transport protocol with RSVP resource reservation to deliver voice application over the network.

Although the UDP transport protocol uses RSVP resource reservation, the performance is not improved significantly. However, by looking at the UDP graph in scenario (1), it can be seen that UDP with the RSVP resource reservation performs better than UDP without RSVP.

The main reason for the better performance of ECTP with RSVP is again the ECTP tree-based architecture, and QoS management functionality, i.e. reserving resources (RSVP), could provide better reliability and scalability (over nodes) along with dynamic admission control compared to the UDP transport protocol. Hence, the traffic delays during the transmission of voice applications can reduce by ECTP transport protocol enabled by the standard RSVP.

7.2.3 SCENARIO (3)- ECTP

In this scenario, all the nodes use either ECTP enabled by RSVP resource reservation or ECTP without RSVP resource reservation. Figure (7.6) illustrates the network model project "*ECTP_RSVP*", scenario (3) "*ECTP*".

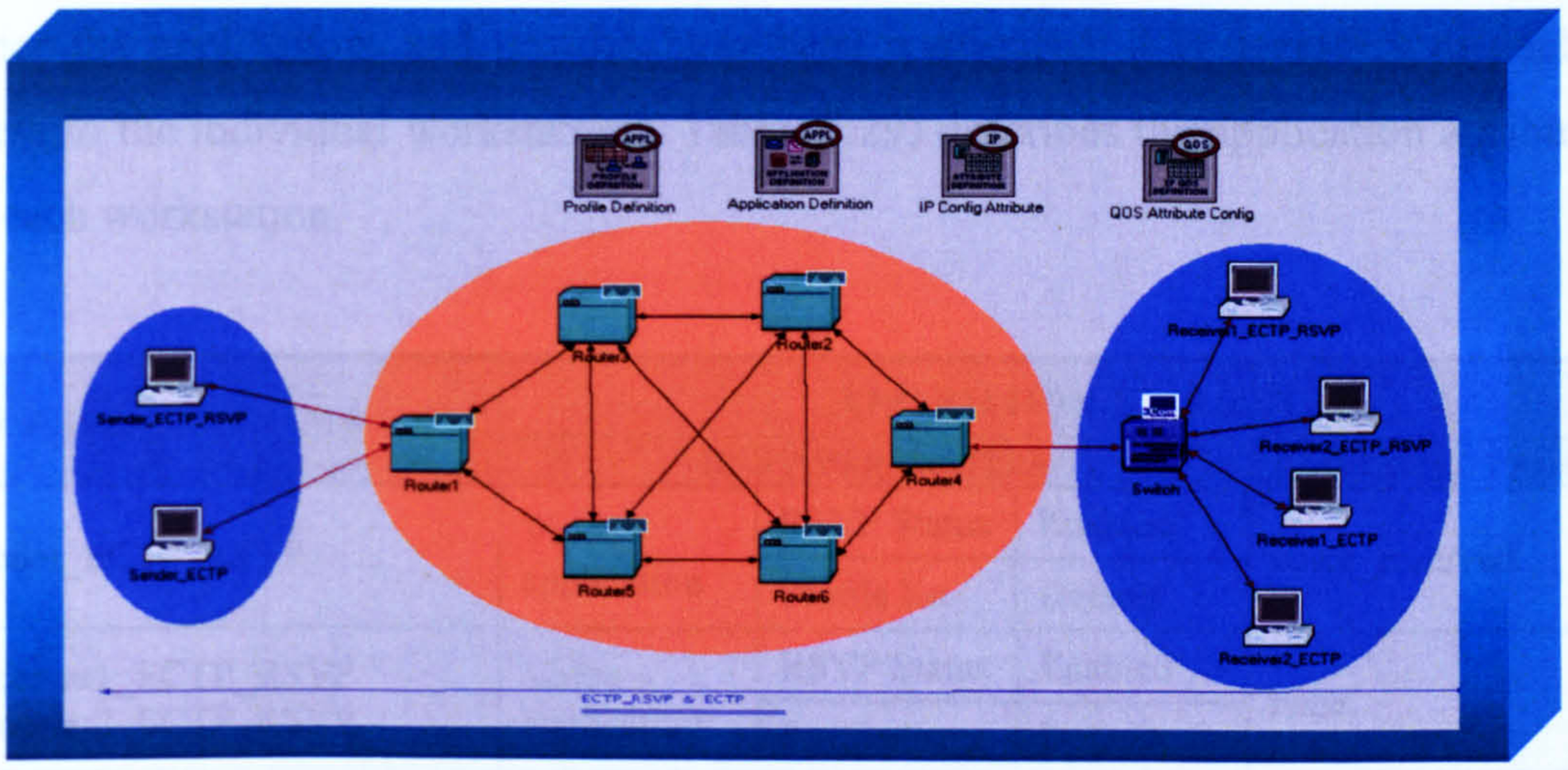


Figure (7.6): OPNET Network model project ECTP_RSVP, Scenario (1)-UDP

7.2.3.1 NETWORK MODEL DESCRIPTION

The network model consists of four components, which are described as follows:

1. **WORKSTATIONS (SENDERS AND RECEIVERS) -** the network model consists of six workstations; *ppp_wksn_adv*. Table (7.28) describes each workstation model and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	ppp_wks_adv_ECTP_RSVP	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management rsvp
Sender_ECTP Receiver1_ECTP Receiver2_ECTP	ppp_wksn_adv_ECTP	ECTP part 1	ECTP_manager ECTP_connection

Table (7.28): Description of workstation models

2. **ROUTERS** - See section (7.2.1.1), Routers.

3. **APPLICATION DEFINITION**- See table (7.11), Application definition.

4. **PROFILE DEFINITION**- See table (7.12), Profile definition.

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.29) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES			
	RSVP PARAMETERS			SUPPORTED PROFILES
Sender_ECTP_RSVP	Voice application	RSVP Status	Enabled	voice_reserved
		Profile list	Default	
Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	Voice application	RSVP Status	Enabled	none
		Profile list	Default	
Sender_ECTP	none			voice_unreserved
Receiver1_ECTP Receiver2_ECTP	none			none

Table (7.29): Description of application attributes

5. **TRAFFIC DESCRIPTIONS**- Traffic between *Sender_ECTP_RSVP*, *Receiver1_ECTP_RSVP* and *Receiver2_ECTP_RSVP* uses Reserved type of service with the integer priority value of 224 and delay parameter. Traffic between *Sender_ECTP*, *Receiver1_ECTP* and *Receiver2_ECTP* uses Standard type of service with the integer priority value of 64 and delay parameter.

6. **IP CONFIGURATION**- See section (7.2.1.1), IP configuration.

7. **RSVP CONFIGURATION**- See section (7.2.1.1), RSVP configuration.

7.2.3.2 COLLECTING STATISTICS

See section (7.2.1.2), Collecting statistics.

7.2.3.3 SIMULATION SETUP

See section (7.2.1.3), simulation setup.

7.2.3.4 SIMULATION RESULTS

Figure (7.7) illustrates the simulation results of end-to-end traffic delay over voice application in ECTP scenario. The ECTP_RSVP (green) graph represents the variations of the traffic delay parameter on the network using the ECTP transport protocol with RSVP. The ECTP (pink) graph represents the variations of the traffic delay parameter on the network using the ECTP transport protocol without RSVP.

- object: ECTP_RSVP of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object: ECTP of logical network
- Annotation: voice_unreserved/Voice RSVP Not Used

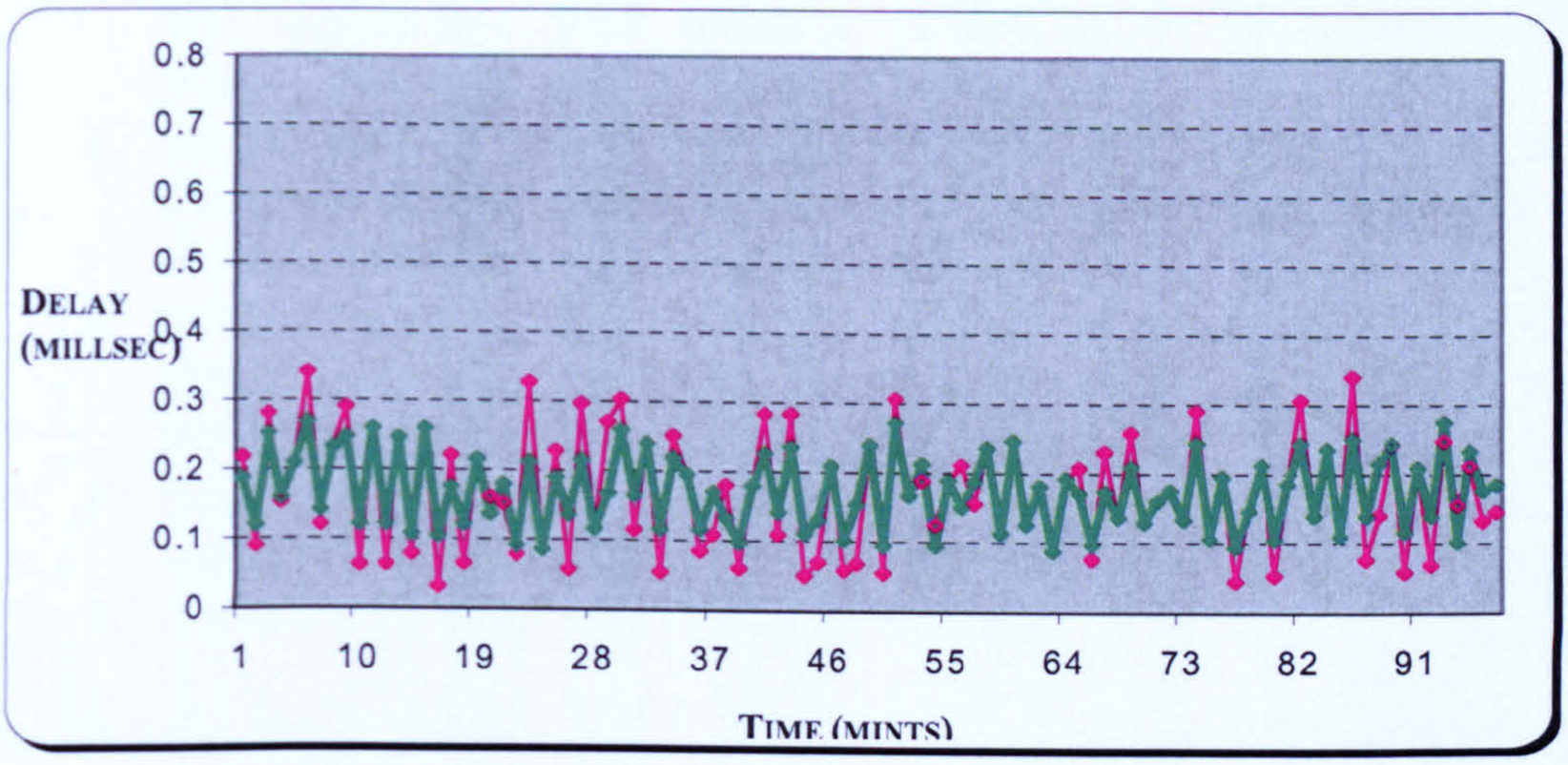


Figure (7.7): Voice application. Packet End-to-End Delay (millsec)

7.2.3.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.7) compares the voice application's traffic delay experienced using ECTP transport protocol with or without RSVP resource reservation. As has been shown in Chapters 4 and 5, the ECTP transport protocol itself cannot guarantee the QoS levels required by applications. RSVP signaling protocol is used to enable QoS management functionality in ECTP.

As it shown in the ECTP_RSVP graph (pink) during the 91 minutes simulation the traffic delay variation remained steady between *0.08419 (millsec)* minimums and *0.2730 (millsec)* maximums. However, by looking at ECTP graph (pink), the traffic delay variation fluctuates between *0.03254 (millsec)* minimums and *0.34082 (millsec)* maximums.

The overall simulation results show that the ECTP transport protocol with the RSVP resource reservation performs better than ECTP transport protocol without RSVP resource reservation to deliver voice application over a network. The main reason is that although ECTP has the tree-based functionality, but it cannot guarantee the QoS levels required by voice applications. Hence, the performance of the ECTP transport protocol with RSVP resource reservation on traffic delay is much steadier than ECTP without RSVP.

7.3 CASE STUDY (III)- ECTP ENABLED BY STANDARD AND AGGREGATED RSVP

Case Study (III) investigates the behavior and performance of ECTP transport protocol enabled by the standard and aggregated RSVP resource reservation (Chapter 5, section 5.5) using voice application. In this case study, the ECTP transport protocol enabled by the standard and aggregated RSVP is compared with ECTP and UDP transport protocols with and without RSVP using voice application. Case study (III)-project *ECTP_RSVP_DiffServ* consists of four scenarios, which are as follow:

- **SCENARIO (1) - UDP:** All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or UDP without RSVP resource reservation over voice applications.
- **SCENARIO (2) - UDP_RSVP:** All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or UDP with RSVP resource reservation over voice applications.
- **SCENARIO (3) - ECTP:** All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or ECTP without RSVP resource reservation over voice applications.
- **SCENARIO (4) - ECTP_RSVP:** All the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or ECTP with RSVP resource reservation over voice applications.

7.3.1 SCENARIO (1)- UDP

In this scenario, all the nodes use either ECTP enabled by the standard and aggregated RSVP resource reservation or UDP without RSVP resource reservation. Figure (7.8) illustrates network model project "*ECTP_RSVP_DiffServ*", scenario (1) "*UDP*".

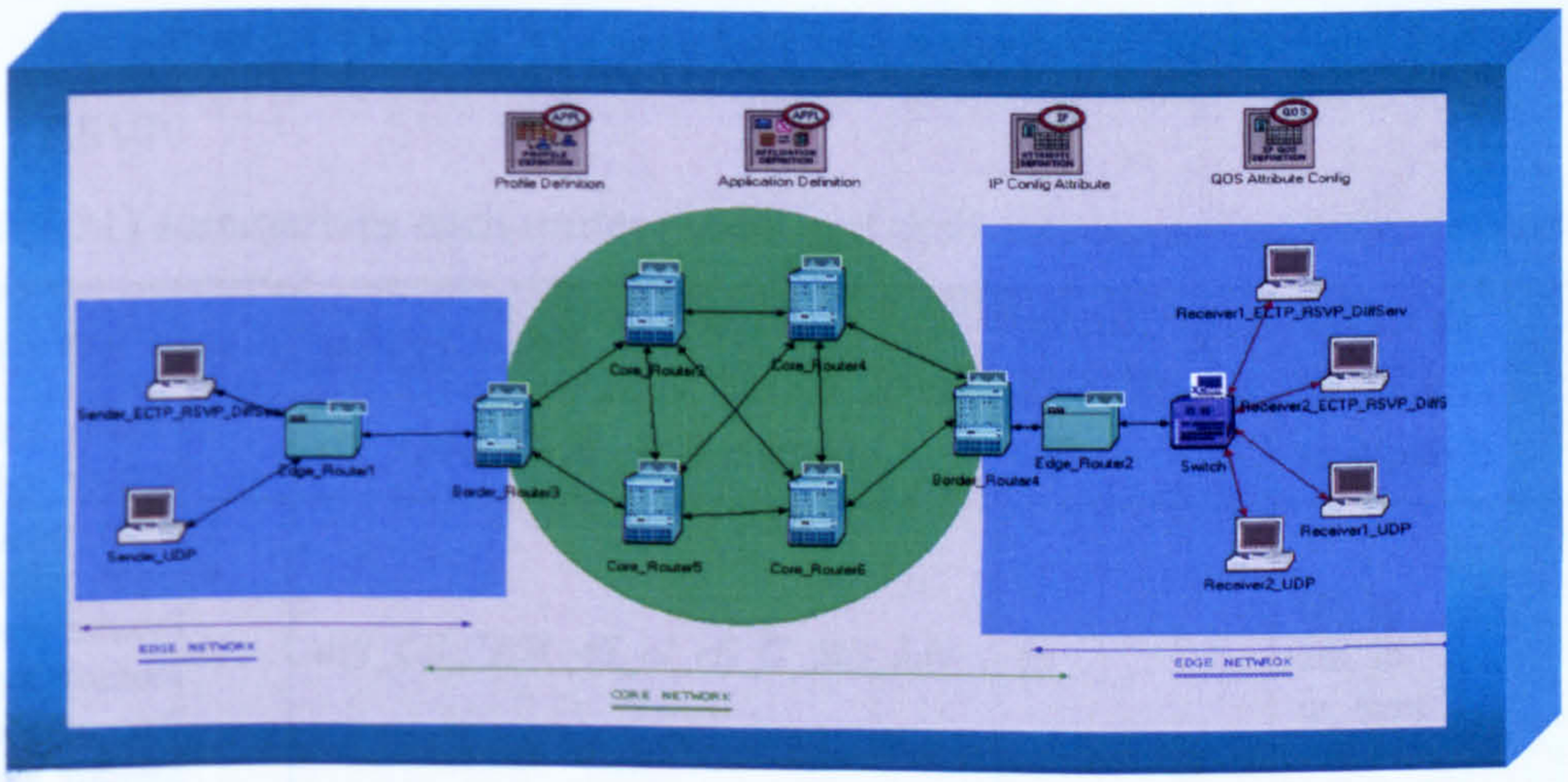


Figure (7.8): OPNET Network model project ECTP_RSVP_DiffServ, Scenario (1)-UDP

7.3.1.1 NETWORK MODEL DESCRIPTION

The network model consists of six components, which are described as follows:

1. **WORKSTATIONS** -the network model consists of six workstations; *ppp_wksn_adv*. Table (7.30) describes each workstation model and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL	PROCESS MODEL
Sender_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	ppp_wksn_adv_ECTP_RSVP_DiffServ	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management aggregated_rsvp
Sender_UDP Receiver1_UDP Receiver2_UDP	ppp_wks_adv_UDP	UDP	rip_udp_v3

Table (7.30): Description of workstation models

2. **ROUTERS**- the network model consists of 2 types of routers, which are described as follow:
   - **EDGE ROUTERS**- there are two edge routers, *CISCO 7204 (CS_7204_4S_al_e8_fl_Sl8_adv)*, which have been chosen randomly to use either ECTP with the standard and aggregated RSVP or UDP without RSVP.
   - **BORDER AND CORE ROUTERS**- There are two border and four core routers, *CISCO 7204 (diff_CS_7204_4S_al_e8_fl_Sl8_adv)* (Chapter 6, section 6.3), which are enabled by DiffServ architecture to deliver aggregated RSVP in core network (Chapter 6, section 6.3). They have been chosen randomly to use either ECTP with the standard and aggregated RSVP or UDP without RSVP.

Table (7.31) summarizes each router model and their corresponding process models.

ROUTER TYPE	MODEL NAME	NODE MODEL	PROCESS MODEL
Edge_router1 Edge_router2	*CS_7204_4S_al_e8_fl_Sl8_adv*	ip	ip_dispatch
Border_router3 Border_router4 Core_router3 Core_router4 Core_router5 Core_router6	*diff_CS_7204_4S_al_e8_fl_Sl8_adv*	ip	DS_ip DS_ip ip_icmp

Table (7.31): Description of Routers models and corresponding process model

3. **APPLICATION DEFINITION-** Table (7.32) describes the application definition parameters in scenario (1)_UDP.

APPLICATION DEFINITION	DESCRIPTION	PARAMETERS	VALUES	
Voice RSVP Not Used	Voice Application	Silence Length (second)	None	
		Talk Spurt length (seconds)	None	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Best Effort	
		RSVP parameters	None	
		Traffic Mix (%)	All Discrete	
Voice RSVP Used	Voice Application	Silence Length (second)	default	
		Talk Spurt length (seconds)	default	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Reserved	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	
Voice Aggregated RSVP Used	Voice Application	Silence Length (second)	Incoming silence length(sec)	Exponential (1.0)
			Outgoing silence length (sec)	Exponential (1.1)
		Talk Spurt length (seconds)	Incoming talk spurt length(sec)	Exponential (1.0)
			Outgoing talk spurt length (sec)	Exponential (1.1)
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	DSCP	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	

Table (7.32): Description of application definition

4. **PROFILE DEFINITION** - Table (7.33) describes the profile definition of scenario (1)_UDP.

PROFILE DEFINITION	PROFILE CONFIGURATION	PARAMETERS	APPLICATION	PARAMETERS
voice_unreserved	Application	Voice RSVP Not used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		
voice_reserved	Application	Voice RSVP used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Once at start time
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		
voice_aggregated_reserved	Application	Voice aggregated RSVP Used	Start time offset (secs)	Uniform (5,10)
			Duration (secs)	End of profile
			Repeatability	Inter-repetition time(sec) Exponential (300)
				No. of repetition unlimited
				Repetition pattern Serial
	Operation mode	Serial (ordered)		
	Start time (sec)	Uniform (100,110)		
	Duration (sec)	End of simulation		
	Repeatability	Once at start time		

Table (7.33): Description of profile definition and configuration

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.34) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES				
	RSVP PARAMETERS			SUPPORTED PROFILES	SUPPORTED PROFILES
Sender_ECTP_RSVP_DiffServ	Voice application	RSVP Status	Enabled	voice_reserved	Supported
		Profile list	Default		
Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	Voice application	RSVP Status	Enabled	none	Supported
		Profile list	Default		
Sender_UDP	none			voice_unreserved	none
Receiver1_UDP Receiver2_UDP	none			none	none

Table (7.34): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.32). Traffic between *Sender_ECTP_RSVP_DiffServ*, *Receiver1_ECTP_RSVP_DiffServ* and *Receiver2_ECTP_RSVP_DiffServ* uses Reserved type of service and Differentiated services with the integer priority value of 252 and delay parameter. The reservation has been made for traffic in both directions. Traffic between *Sender_UDP*, *Receiver1_UDP* and *Receiver2_UDP* uses Best effort type of service with the integer priority value of 0 and delay parameter.

6. **IP CONFIGURATION -** Table (7.35) defines the IP configuration attributes in scenario (1) _UDP.

IP CONFIGURATION ATTRIBUTE	VALUE	
IP compression Information	Standard compression	
IP multicast RP Information	Default	
IP Ping parameters	Interval (sec)	1.0
	Packet size (bytes)	64
	Count	5
	Timeout (sec)	20
	Record route	Enabled
IP Route table export	Status	Enabled
	Export Time(s) specification	Once at end of simulation

Table (7.35): Description of IP configuration attributes

7. **RSVP CONFIGURATION-** See section (7.2.1.1), RSVP configuration.

7.3.1.2 COLLECTING STATISTICS

Table (7.36) defines the collected statistics in scenario (1)-UDP.

STATISTICS	COLLECTION	PARAMETERS
Global Statistics	IP	Traffic dropped (packet/sec)
	RSVP	Control traffic received (packet/sec)
		Control traffic sent (packet/sec)
Node Statistics	RSVP	Path messages received (packet/sec)
		Path messages sent(packet/sec)
		Resv Conf messages received (packet/sec)
		Resv Conf messages sent(packet/sec)
		Resv messages received (packet/sec)
		Resv messages sent (packet/sec)
	IP Interface	Queue delay variation (sec)
		RSVP allocated bandwidth (bytes/sec)
	Voice application	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice called party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)
	Voice calling party	Packet end_to_end delay (sec)
		Traffic Received (bytes/sec)
		Traffic Received (packets/sec)
		Traffic sent (bytes/sec)
		Traffic sent (packet/sec)

Table (7.36): Collected statistics in network model

7.3.1.3 SIMULATION SETUP

Before running the simulation, OPNET will set up values for simulation attributes. Simulation duration is set to 1.0 hour. Table (7.37) describes the simulation setup attributes for this network model.

SIMULATION SETUP ATTRIBUTES	VALUE
APR Sim Efficiency	Enabled
BGP Sim Efficiency mode	Enabled
IGM Sim Efficiency	Enabled
IP Dynamic Routing Protocol	Default
IP Routing Table	Import/Export
OSPF Sim Efficiency	Enabled
RIP Sim Efficiency	Enabled
RSVP Sim Efficiency	Enabled

Table (7.37): Simulation setup Attributes

7.3.1.4 SIMULATION RESULTS

Figure (7.9) illustrates the simulation results of end-to-end traffic delay over voice application in UDP scenario. The ECTP_RSVP_DiffServ (turquoise) graph represents the variations of the traffic delay parameter on the network using ECTP transport protocol with the standard and aggregated RSVP. The UDP (blue) graph represents the variation of traffic delay parameter on the network using UDP transport protocol without RSVP.

- object: ECTP_RSVP_DiffServ of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object: UDP of logical network
- Annotation: voice_unreserved/Voice RSVP Not Used

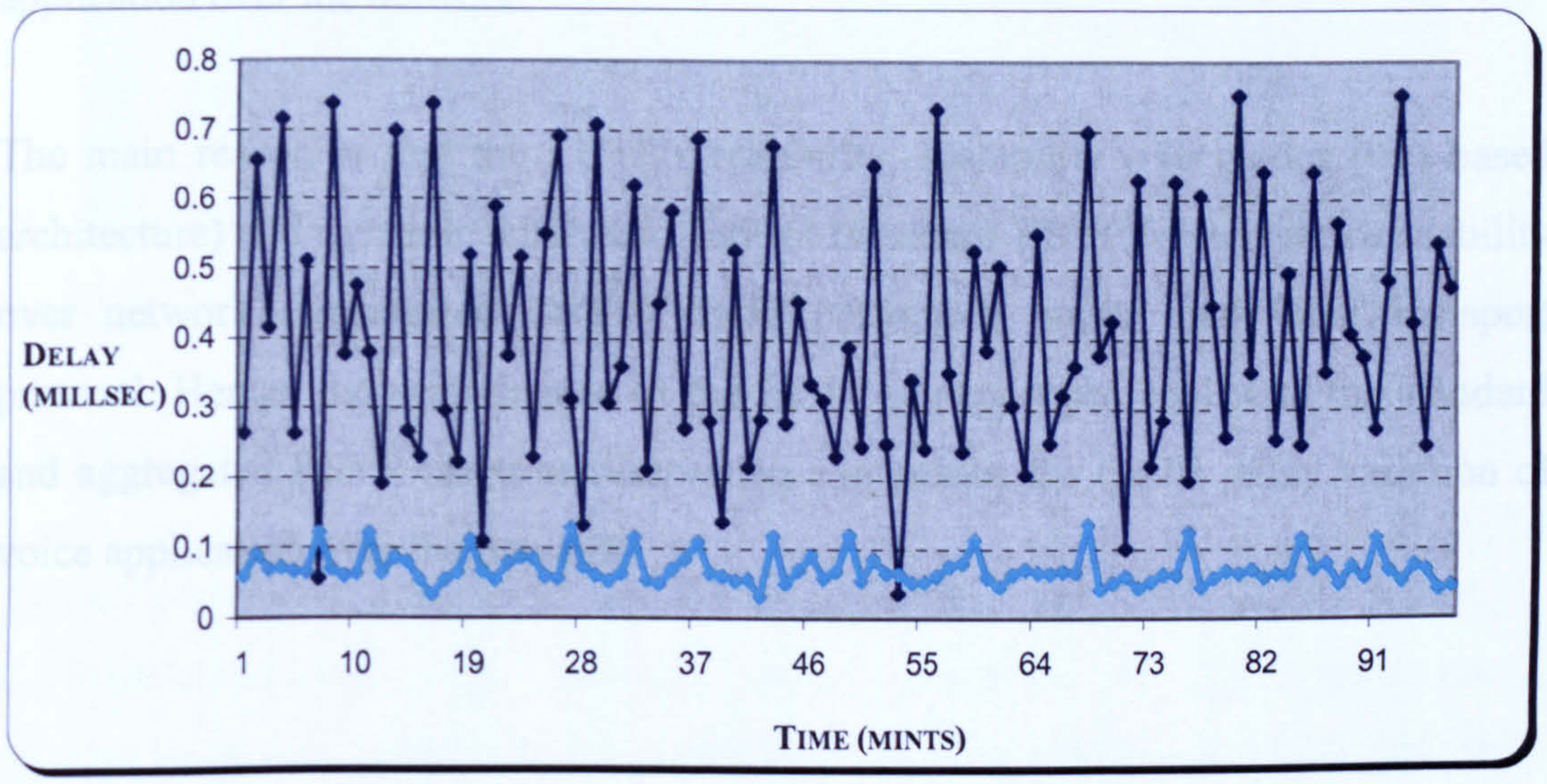


Figure (7.9): Voice application. Packet End-to-End Delay (millsec)

7.3.1.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.9) compares the voice application's traffic delay experienced using ECTP transport protocol with the standard and aggregated RSVP resource reservation and UDP transport protocol without RSVP resource reservation. As mentioned in Chapter 3, section (3.8), the benefit of interoperation between IntServ with standard RSVP and DiffServ with aggregated RSVP provides dynamic admission control and efficient network resources by IntServ, along with scalability by DiffServ over core networks. Also as it was proposed in Chapter 5, section (5.5), ECTP QoS parameters

should be mapped in to the standard RSVP and then in to the aggregated RSVP traffic descriptors.

As it shown in the ECTP graph (turquoise) during the 91 seconds simulation the traffic delay variation stayed steady between *0.03028* (*millsec*) minimums and *0.1350* (*millsec*) maximums. However, by looking at the UDP graph (blue) the traffic delay variation fluctuates between *0.056* (*millsec*) minimums and *0.751* (*millsec*) maximums.

The overall simulation results show that the ECTP transport protocol with the standard and aggregated RSVP resource reservation performs better than UDP transport protocol without RSVP resource reservation when delivering voice application over the network.

The main reason is that the ECTP's reliability, scalability over nodes (tree-based architecture) and dynamic admission control (standard RSVP) along with scalability over network (aggregated RSVP) could performed better than UDP transport protocol. Hence, the performance of the ECTP transport protocol with the standard and aggregated RSVP resource reservation can reduce the traffic delay variation of voice application over the network.

7.3.2 SCENARIO (2)- UDP_RSVP

In this scenario, all the nodes use either ECTP enabled by standard and aggregated RSVP resource reservation or UDP with RSVP resource reservation. Figure (7.10) illustrates network model project ECTP_RSVP_DiffServ, scenario (2).

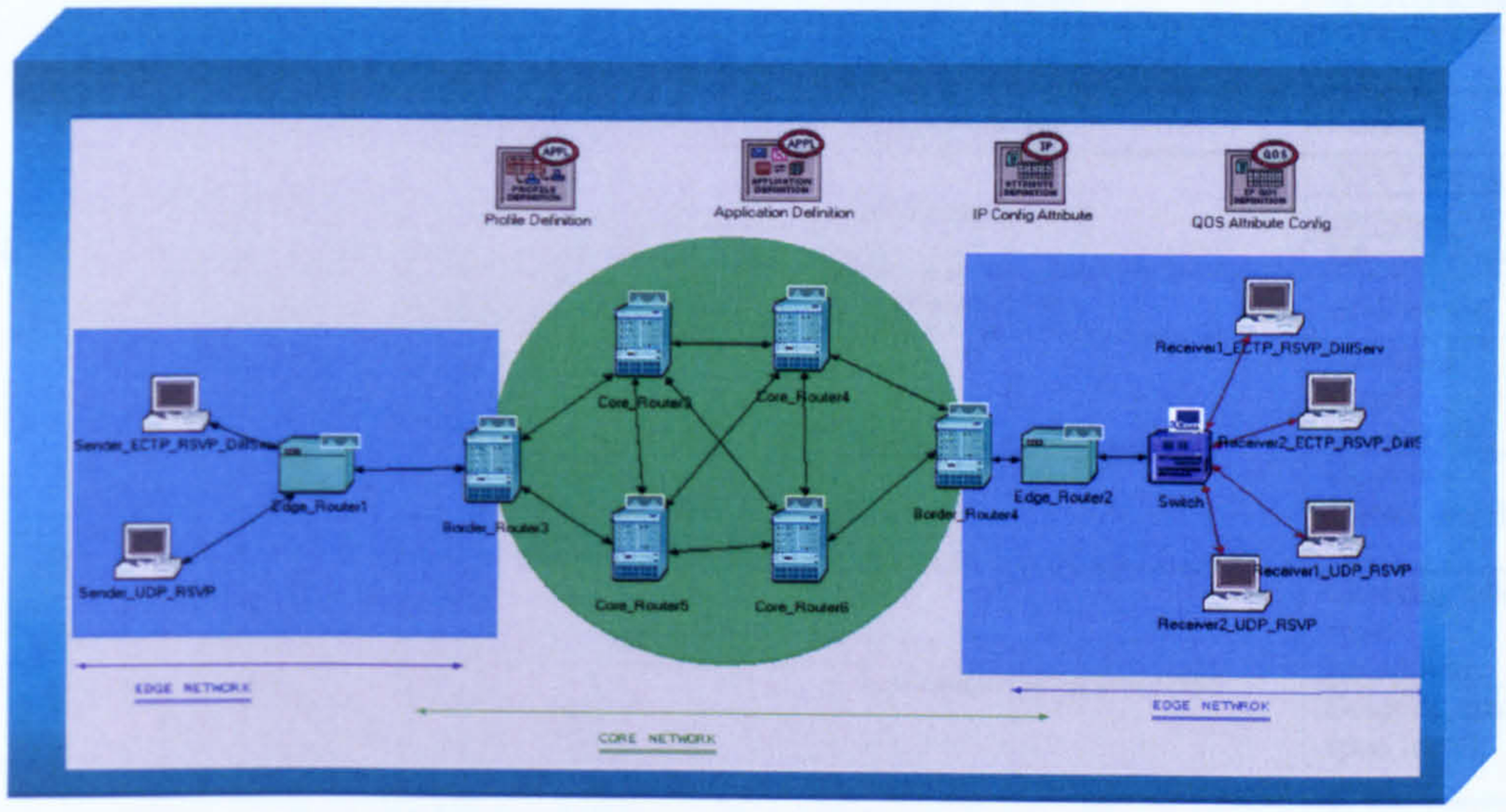


Figure (7.10): OPNET Network model project ECTP_RSVP_DiffServ, Scenario (1)-UDP_RSVP

7.3.2.1 NETWORK MODEL DESCRIPTION

This network model consists of four components, which are described as follows:

1. **WORKSTATIONS** -the network model consists of six workstations, *ppp_wksn_adv*. Table (7.38) describes the workstation models and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	ppp_wksn_adv_ECTP_RSVP_DiffServ	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management aggregated_rsvp
Sender_UDP_RSVP Receiver1_UDP_RSVP Receiver2_UDP_RSVP	ppp_wksn_adv_UDP_RSVP	UDP	rip_udp_v3 rsvp

Table (7.38): Description of workstation models

2. **ROUTERS**- See section (7.3.1.1), Routers.

3. **APPLICATION DEFINITION-** Table (7.39) describes the application definition parameters in scenario (2), UDP_RSVP.

APPLICATION DEFINITION	DESCRIPTION	PARAMETERS	VALUES	
Voice RSVP Used	Voice Application	Silence Length (second)	default	
		Talk Spurt length (seconds)	default	
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	Standard	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	
Voice Aggregated RSVP Used	Voice Application	Silence Length (second)	Incoming silence length(sec)	Exponential (1.0)
			Outgoing silence length (sec)	Exponential (1.1)
		Talk Spurt length (seconds)	Incoming talk spurt length(sec)	Exponential (1.0)
			Outgoing talk spurt length (sec)	Exponential (1.1)
		Symbolic Destination Name	Voice Receiver	
		Encoder Scheme	G.711(silence)	
		Voice Frames per Packet	1	
		Type of service	DSCP	
		RSVP parameters	RSVP Statue	Enabled
			outbound Flow	Default
			Inbound Flow	Default
		Traffic Mix (%)	All Discrete	

Table (7.39): Description of application definition

4. **PROFILE DEFINITION-** Table (7.40) describes the profile definition of scenario (2), UDP_RSVP.Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.41) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES				
	RSVP PARAMETERS			SUPPORTED PROFILES	SUPPORTED PROFILES
Sender_ECTP_RSVP_DiffServ Sender_UDP_RSVP	Voice application	RSVP Status	Enabled	voice_reserved	Supported
		Profile list	Default		
Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ Receiver1_UDP_RSVP Receiver2_UDP_RSVP	Voice application	RSVP Status	Enabled	none	Supported
		Profile list	Default		

Table (7.40): Description of profile definition and configuration

Table (7.41): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.40). Traffic between *Sender_ECTP_RSVP_DiffServ*, *Receiver1_ECTP_RSVP_DiffServ* and *Receiver2_ECTP_RSVP_DiffServ* uses Reserved type of service and Differentiated services with the integer priority value of 252 and delay parameter. Traffic between *Sender_UDP_RSVP*, *Receiver1_UDP_RSVP* and *Receiver2_UDP_RSVP* uses Reserved type of service with the integer priority value of 224 and delay parameter. The reservation has been made for traffic in both directions.

6. **IP CONFIGURATION-** See section (7.3.1.1), IP configuration.

7. **RSVP CONFIGURATION-** See section (7.2.1.1), RSVP configuration.

7.3.2.2 COLLECTING STATISTICS

See section (7.3.1.2), collecting statistics.

7.3.2.3 SIMULATION SETUP

See section (7.3.1.3), simulation setup.

7.3.2.4 SIMULATION RESULTS

Figure (7.11) illustrates the simulation results of end-to-end traffic delay over voice application in UDP_RSVP scenario. The ECTP_RSVP_DiffServ (turquoise) graph represents the variations of the traffic delay parameter on the network using ECTP transport protocol with the standard and aggregated RSVP. The UDP (red) graph represents the variation of the traffic delay parameter on the network using UDP transport protocol with RSVP.

- object: ECTP_RSVP_DiffServ of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object: UDP_RSVP of logical network
- Annotation: voice_reserved/Voice RSVP Used

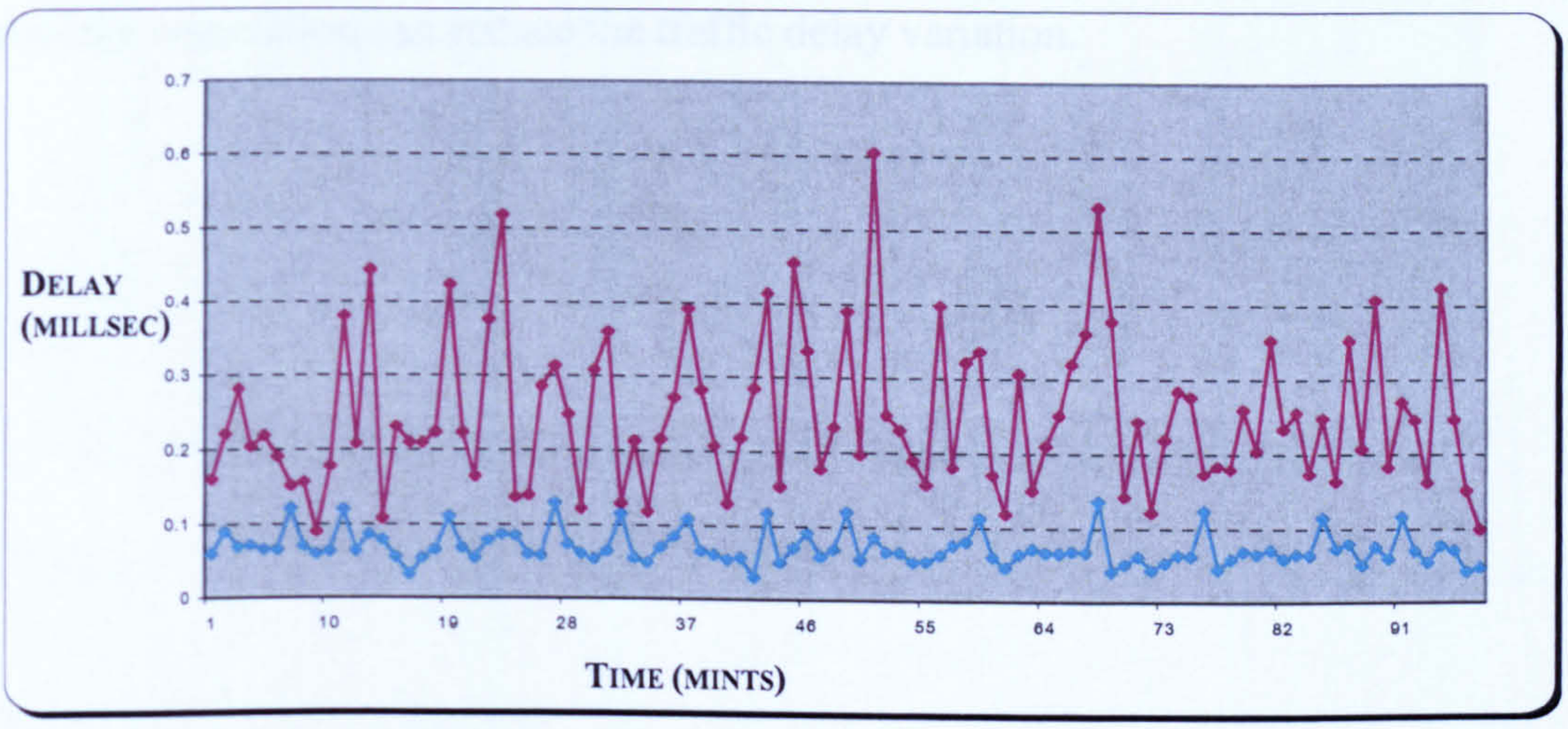


Figure (7.11): Voice application. Packet End-to-End Delay (millsec)

7.3.2.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.11) compares the voice application's traffic delay experienced using ECTP transport protocol with the standard and aggregated RSVP resource reservation and UDP transport protocol with RSVP resource reservation.

As shown in the ECTP graph (turquoise), during the 91 minutes simulation the traffic delay variation stayed steady between *0.03028* (*millsec*) minimums and *0.1350* (*millsec*) maximums. However, by looking at the UDP graph (blue) the traffic delay variation fluctuates between *0.0897* (*millsec*) minimums and *0.6046* (*millsec*) maximums.

The overall simulation results show that the ECTP transport protocol with the standard and aggregated RSVP resource reservation performs better than UDP transport protocol with RSVP resource reservation to deliver voice application over network. Although UDP transport protocol uses RSVP resource reservation, the performance is not significantly improved.

The main reason is that the ECTP's reliability, scalability over nodes (tree-based architecture) and dynamic admission control (standard RSVP) along with scalability over network (aggregated RSVP) could performed better than the UDP transport protocol. Hence, ECTP transport protocol with standard and aggregated RSVP resource reservation can reduce the traffic delay variation.

7.3.3 SCENARIO (3)- ECTP

In this scenario, all the nodes use either ECTP enabled by the standard and aggregated RSVP resource reservation or ECTP without RSVP resource reservation. Figure (7.12) illustrates network model project "*ECTP_RSVP_DiffServ*", scenario (3) "*ECTP*".

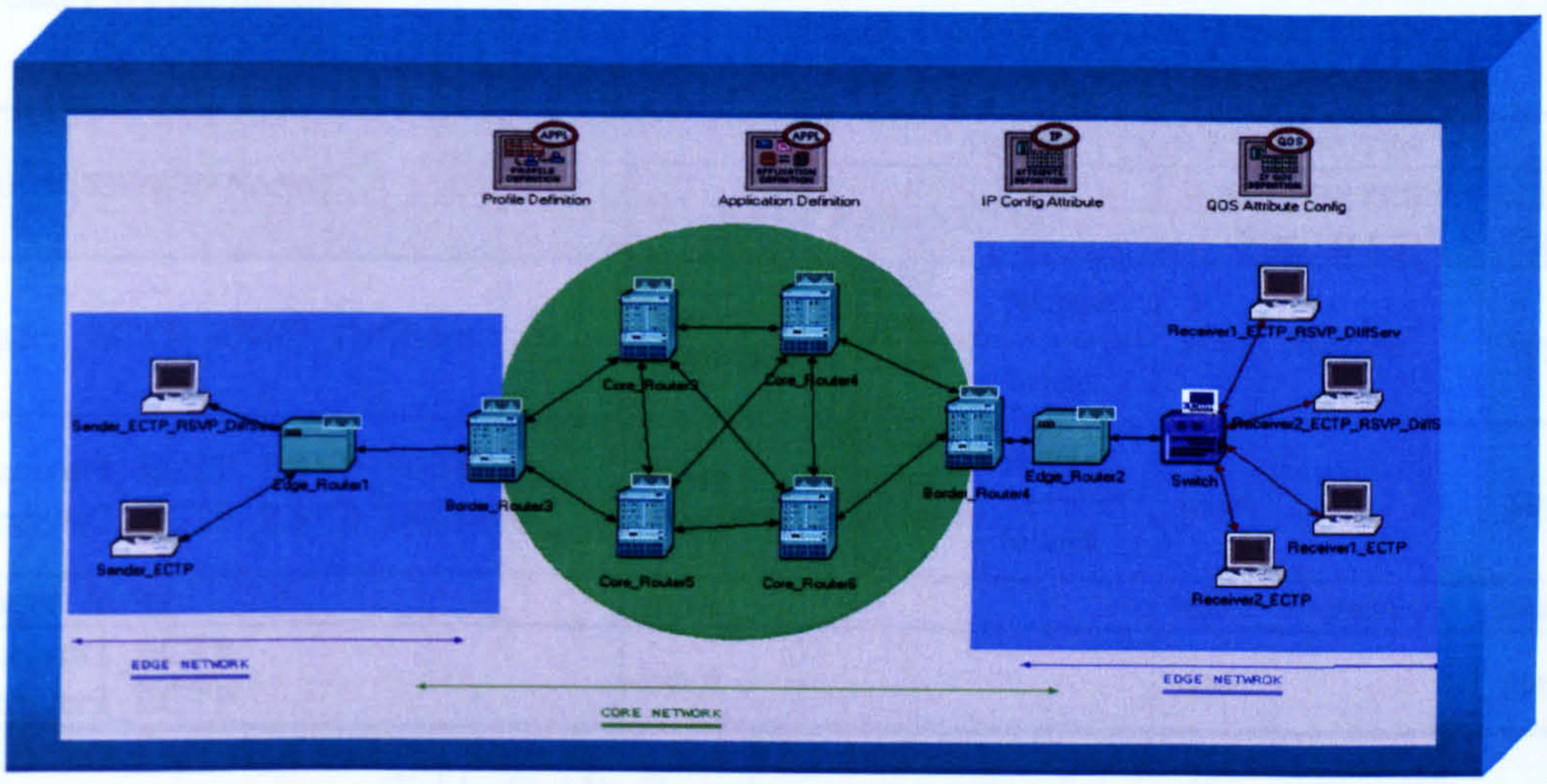


Figure (7.12): OPNET Network model project ECTP_RSVP_DiffServ, Scenario (1)-ECTP

7.3.3.1 NETWORK MODEL DESCRIPTION

The network model consists of four components, which are described as follows:

1. **WORKSTATIONS** -the network model consists of six workstations; *ppp_wksn_adv*. Table (7.42) describes the workstation models and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	ppp_wksn_adv_ECTP_ RSVP_DiffServ	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management aggregated_rsvp
Sender_ECTP Receiver1_ECTP Receiver2_ECTP	ppp_wksn_adv_ECTP	ECTP part 1	ECTP_manager ECTP_connection

Table (7.42): Description of workstation models

2. **ROUTERS-** See section (7.3.1.1), Routers.

3. **APPLICATION DEFINITION-** See table (7.32), application definition.

4. **PROFILE DEFINITION-** See table (7.33), profile definition.

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.43) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES				
	RSVP PARAMETERS			SUPPORTED PROFILES	SUPPORTED PROFILES
Sender_ECTP_RSVP_DiffServ	Voice application	RSVP Status	Enabled	voice_reserved	Supported
		Profile list	Default		
Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	Voice application	RSVP Status	Enabled	none	Supported
		Profile list	Default		
Sender_ECTP	none			voice_unreserved	none
Receiver1_ECTP Receiver2_ECTP	none			none	none

Table (7.43): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.32). Traffic between *Sender_ECTP_RSVP_DiffServ*, *Receiver1_ECTP_RSVP_DiffServ* and *Receiver2_ECTP_RSVP_DiffServ* uses Reserved type of service and Differentiated services with the integer priority value of 252 and delay parameter. The reservation has been made for traffic in both directions. Traffic between *Sender_ECTP*, *Receiver1_ECTP* and *Receiver2_ECTP* uses Standard type of service with the integer priority value of 64 and delay parameter.

6. **IP CONFIGURATION-** See section (7.3.1.1), IP configuration.

7. **RSVP CONFIGURATION-** See section (7.2.1.1), RSVP configuration.

7.3.3.2 COLLECTING STATISTICS

See section (case study (III)-collecting statistics, scenario (1)-UDP)

7.3.3.3 SIMULATION SETUP

See section (7.3.1.2), simulation setup.

7.3.3.4 SIMULATION RESULTS

Figure (7.13) illustrates the simulation results of end-to-end traffic delay over voice application in ECTP scenario. The ECTP_RSVP_DiffServ (turquoise) graph represents the variations of traffic delay parameter on the network using ECTP transport protocol with standard and aggregated RSVP. The ECTP (pink) graph represents the variations of traffic delay parameter on the network using ECTP transport protocol without RSVP.

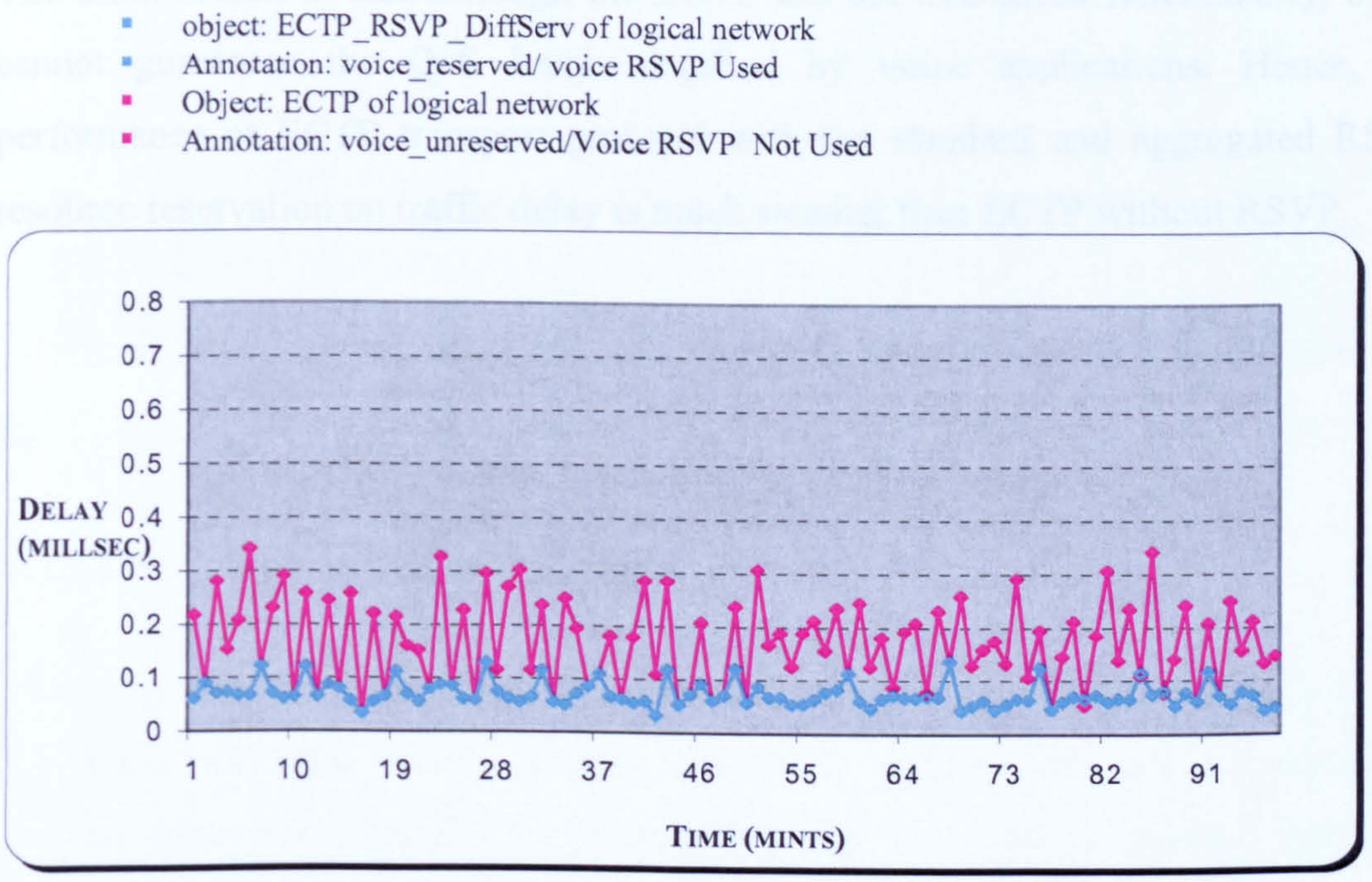


Figure (7.13): Voice application. Packet End-to-End Delay (millsec)

7.3.3.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.13) compares the voice application's traffic delay experienced using ECTP transport protocol with the standard and aggregated RSVP resource reservation and ECTP transport protocol without RSVP resource reservation. As it has mentioned, ECTP transport protocol itself cannot guarantee the QoS levels required by applications.

As shown on the ECTP graph (turquoise) during the 91 seconds simulation the traffic delay variation stayed steady between *0.03028* (*millsec*) minimums and *0.1350* (*millsec*) maximum. However, by looking at ECTP graph (pink), the traffic delay variation fluctuates between *0.03254* (*millsec*) minimums and *0.34082* (*millsec*) maximum.

The overall simulation results show that the ECTP transport protocol with the standard and aggregated RSVP resource reservation performs better than ECTP transport protocol without RSVP resource reservation to deliver voice application over network.

The main reason is that although the ECTP has the tree-based functionality, but it cannot guarantee the QoS levels required by voice applications. Hence, the performance of ECTP transport protocol with the standard and aggregated RSVP resource reservation on traffic delay is much steadier than ECTP without RSVP.

7.3.4 SCENARIO (4)- ECTP_RSVP

In this scenario, all the nodes use either ECTP enabled by the standard and aggregated RSVP resource reservation or ECTP with RSVP resource reservation. Figure (7.14) illustrates network model project "*ECTP_RSVP_DiffServ*", scenario (4) "*ECTP_RSVP*".

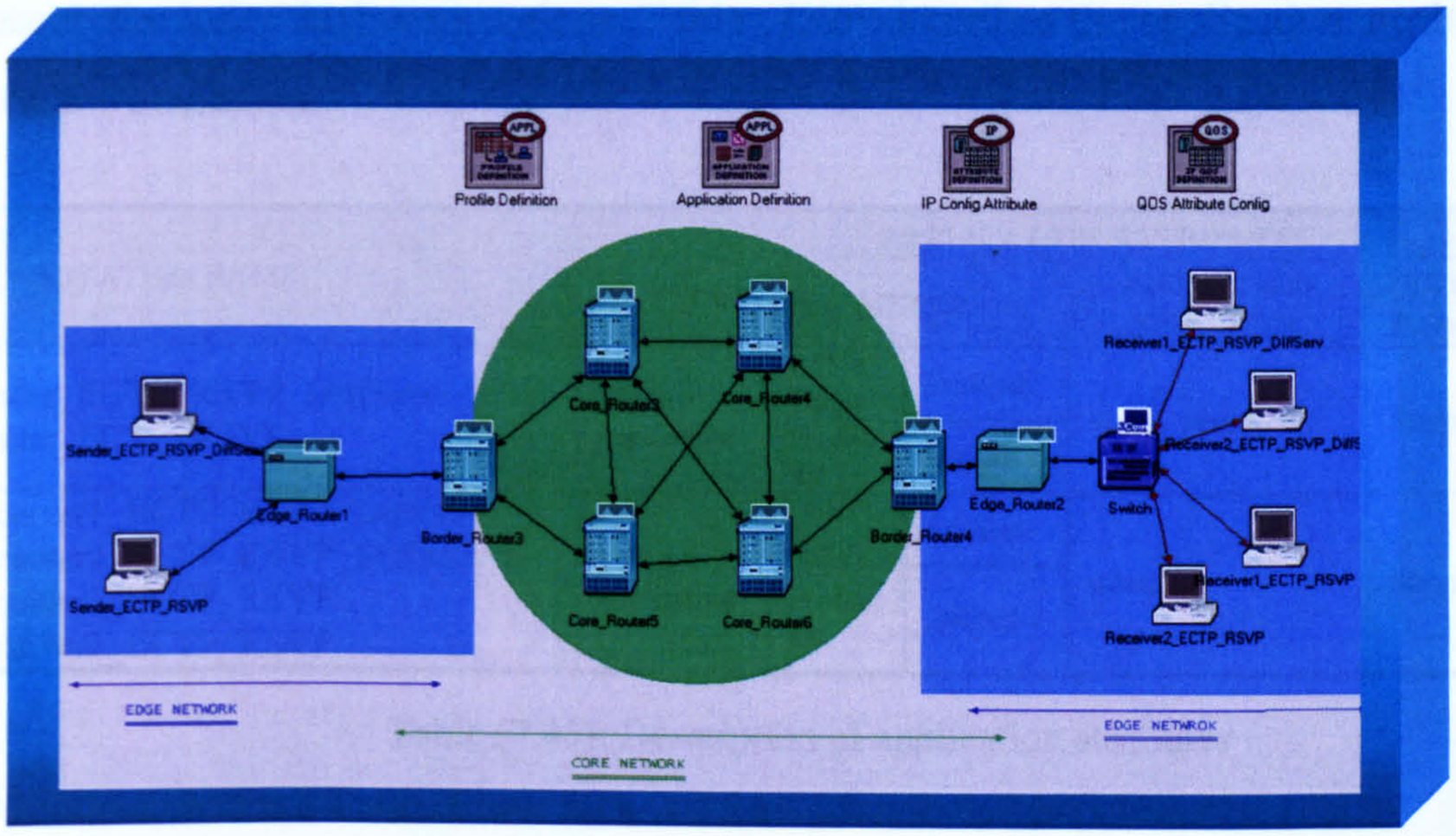


Figure (7.14): OPNET Network model project ECTP_RSVP_DiffServ, Scenario (1)-ECTP_RSVP

7.3.4.1 NETWORK MODEL DESCRIPTION

This network model consists of four components, which are described as follows:

1. **WORKSTATIONS** -the network model consists of six workstations; *ppp_wksn_adv*. Table (7.45) describes the workstation models and their corresponding process models.

WORKSTATION	WORKSTATION MODEL	TRANSPORT PROTOCOL USED IN NODE MODEL	PROCESS MODEL
Sender_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	ppp_wksn_adv_ECTP_RSVP_DiffServ	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management rsvp aggregated_rsvp
Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	ppp_wks_adv_ECTP_RSVP	ECTP part 2 (ECTP_RSVP)	ECTP_manager ECTP_connection ectp_QoS_management rsvp

Table (7.44): Description of workstation models

2. **ROUTERS-** See section (7.3.1.1), Routers.

3. **APPLICATION DEFINITION-** See table (7.39), application definition.

4. **PROFILE DEFINITION-** See table (7.40), profile definition.

Once the applications and profiles have been configured, it is necessary to deploy these to the individual workstations. Table (7.46) describes the application attributes in each workstation.

WORKSTATION NAME	APPLICATION ATTRIBUTES				
	RSVP PARAMETERS			SUPPORTED PROFILES	SUPPORTED PROFILES
Sender_ECTP_RSVP_DiffServ Sender_ECTP_RSVP	Voice application	RSVP Status	Enabled	voice_reserved	Supported
		Profile list	Default		
Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	Voice application	RSVP Status	Enabled	none	Supported
		Profile list	Default		

Table (7.45): Description of application attributes

5. **TRAFFIC DESCRIPTIONS-** This traffic has been specified inside the voice application definition (Table 7.32). Traffic between *Sender_ECTP_RSVP_DiffServ*, *Receiver1_ECTP_RSVP_DiffServ* and *Receiver2_ECTP_RSVP_DiffServ* uses Reserved type of service and Differentiated services with the integer priority value of 252 and delay parameter. Traffic between *Sender_ECTP_RSVP*, *Receiver1_ECTP_RSVP* and *Receiver2_ECTP_RSVP* uses Reserved type of service with the integer priority value of 224 and delay parameter. The reservation has been made for traffic in both directions.

6. **IP CONFIGURATION-** See section (7.3.1.1) IP configuration.

7. **RSVP CONFIGURATION-** See section (7.2.1.1), RSVP configuration.

7.3.4.2 COLLECTING STATISTICS

See section (7.3.1.2), collecting statistics.

7.3.4.3 SIMULATION SETUP

See section (7.3.1.3) simulation setup.

7.3.4.4 SIMULATION RESULTS

Figure (7.15) illustrates the simulation results of end-to-end traffic delay over voice application in ECTP_RSVP scenario. The ECTP_RSVP_DiffServ (turquoise) graph represents the variations of the traffic delay parameter on the network using ECTP transport protocol with the standard and aggregated RSVP. The ECTP_RSVP (green) graph represents the variations of traffic delay parameter on the network using ECTP transport protocol with RSVP.

- object: ECTP_RSVP_DiffServ of logical network
- Annotation: voice_reserved/ voice RSVP Used
- Object:ECTP_RSVP of logical network
- Annotation: voice_reserved/Voice RSVP Used

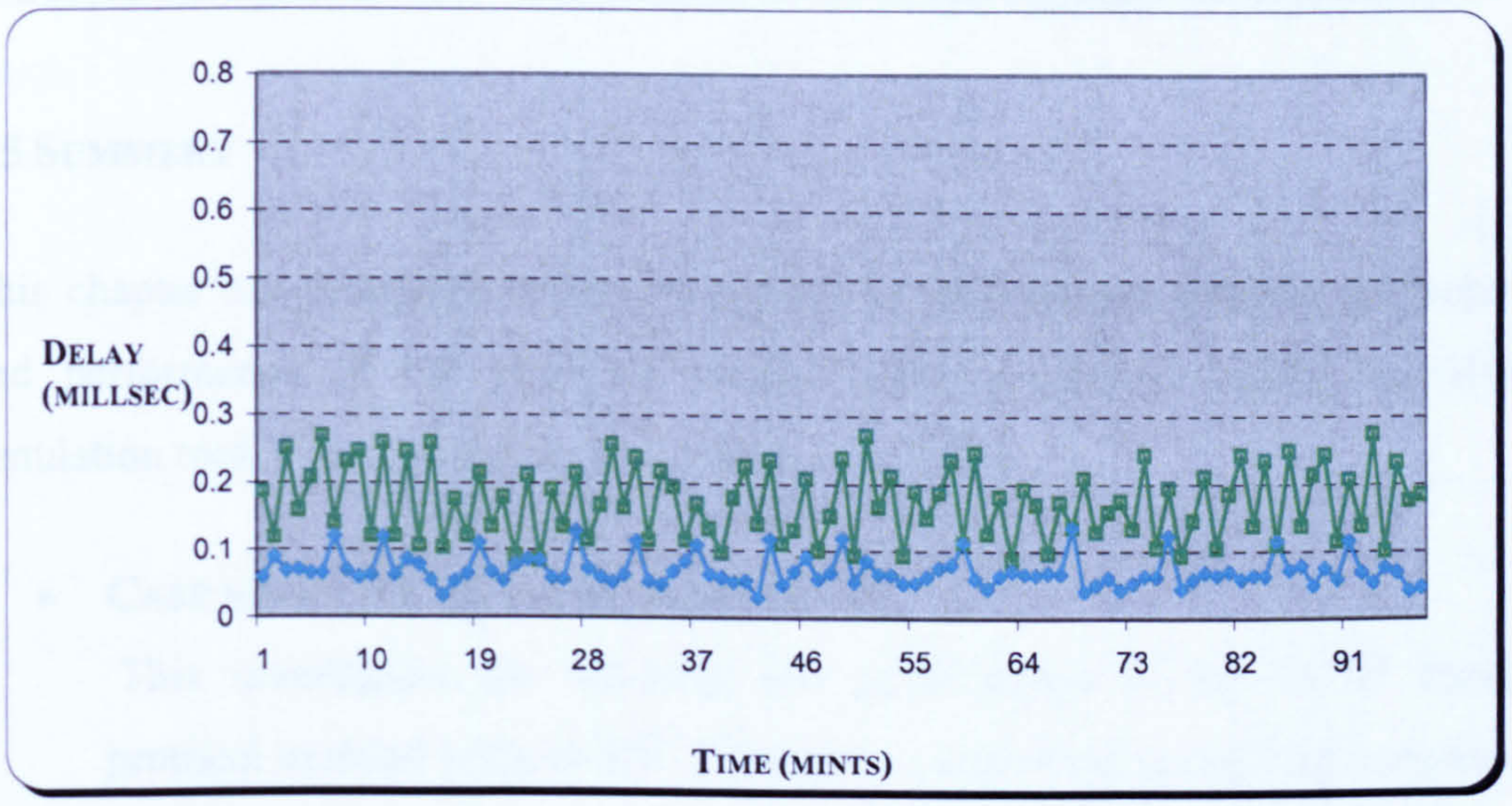


Figure (7.15): Voice application. Packet End-to-End Delay (millsec)

7.3.4.5 ANALYSIS OF TRAFFIC DELAY

Figure (7.15) compares the voice application's traffic delay experienced using ECTP transport protocol with the standard and aggregated RSVP resource reservation and ECTP transport protocol with RSVP resource reservation.

As shown in the ECTP graph (turquoise) during the 91 minutes simulation the traffic delay variation stayed steady between *0.03028* (*millsec*) minimums and *0.1350* (*millsec*) maximums. However, by looking at ECTP graph (pink) traffic delay variation stayed steady between *0.08419* (*millsec*) minimum and *0.2730* (*millsec*) maximums.

The main reason is that the ECTP's reliability, scalability over nodes (tree-based architecture) and dynamic admission control (standard RSVP) along with scalability over network (aggregated RSVP) could performed better than ECTP transport protocol with RSVP resource reservation. Hence, ECTP transport protocol with the standard and aggregated RSVP resource reservation can reduce the traffic delay variation of voice application over the network. In the standard RSVP, per-flow reservation, may require a non-trivial amount of message exchange, computation, and memory resource in each router along the way. It would be better to reduce this to a more manageable level where load is heaviest and aggregation is possible.

7.5 SUMMERY

This chapter has discussed three case studies to simulate and analyze the behavior and performance of the proposed models over a network model in OPNET simulation tool. The case studies are defined as follows:

- **CASE STUDY (I): ECTP WITHOUT RSVP**
  This investigates the behavior and performance of the ECTP transport protocol enabled without RSVP resource reservation using voice application traffic.

- **CASE STUDY (II): ECTP WITH STANDARD RSVP**
  This investigates the behavior and performance of the ECTP transport protocol enabled by standard RSVP resource reservation using voice application traffic.

- **CASE STUDY (I): ECTP WITH STANDARD AND AGGREGATED RSVP**
  This investigates the behavior and performance of the ECTP transport protocol enabled by standard and aggregated RSVP resource reservation using voice application traffic.

The experienced simulation results and analysis for all three case studies has shown that the ECTP transport protocol with the standard and aggregated RSVP resource reservation has the best performance on traffic delay to deliver voice application over the network. As has been discussed, the main reason is that the ECTP's reliability, scalability over nodes (tree-based architecture) and dynamic admission control (standard RSVP) along with scalability over network (aggregated RSVP).

Explicit traffic has been used (appendix 7) for all of case studies in this chapter (sections: 7.1.1, 7.2.1.1, 7.2.2.1, 7.2.3.1, 7.3.1.1, 7.3.2.1, 7.3.3.1, 7.3.4.1 on 3.application definitions, 4. profile definitions and 5. traffic descriptions. The explicit traffic inside OPNET simulator creates application traffic which includes a set of models for generating traffic based on standard applications such as FTP, HTTP and voice. Explicit traffic modelling provides the most accurate results, since it models all protocols effect. However, this also results in longer simulations and higher memory usage application traffic which includes a set of models for generating traffic. This traffic has been specified inside the applications such as HTTP, FTP or voice. Type of service assigned to the packets sent from the both senders. It represents a session attribute which allows packets to be processed faster in ip queues (it is an integer between 0-252, 252 being the highest priority). Table (7.45) shows the traffic descriptions in each case study.

PROJECT	SCENARIO	Traffic between nodes	Descriptions
Without_RSVP	ECTP_UDP	Sender_ECTP Receiver1_ECTP Receiver2_ECTP	uses voice application traffic with standard type of service with priority value of 64 and delay parameter..
		Sender_UDP Receiver1_UDP Rececier2_UDP	uses voice application traffic with Best Effort type of service with priority value of 0 and delay parameter.
ECTP_RSVP	UDP	Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	uses voice application traffic with Reserved type of service with priority value of 224 and delay parameter..
		Sender_UDP Receiver1_UDP Receiver 2_UDP	uses voice application traffic with Best Effort type of service with priority value of 0 and delay parameter.
	UDP_RSVP	Sender_UDP_RSVP Receiver1_UDP Receiver2_UDP	uses voice application traffic with Reservation type of service with priority value of 208 and delay parameter.
	ECTP	Sender_ECTP Receiver1_ECTP Receiver2_ECTP	uses voice application traffic with standard type of service with priority value of 64 and delay parameter..
ECTP_RSVP_DiffServ	UDP	Sender_ECTP_RSVP_DiffServ Receiver1_ECTP_RSVP_DiffServ Receiver2_ECTP_RSVP_DiffServ	uses voice application traffic with Reserved type of service and Differentiated service with priority value of 252 and delay parameter.
		Sender_UDP Receiver1_UDP Rececier2_UDP	uses voice application traffic with Best Effort type of service with priority value of 0 and delay parameter.
	UDP_RSVP	Sender_UDP Receiver1_UDP Rececier2_UDP	uses voice application traffic with Reserved type of service with priority value of 208 and delay parameter.
	ECTP_RSVP	Sender_ECTP_RSVP Receiver1_ECTP_RSVP Receiver2_ECTP_RSVP	uses voice application traffic with Reserved type of service with priority value of 224 and delay parameter..

Chapter 8

Conclusion and Future Work

8. CONCLUSION

The main objective of this research was to develop a framework to provide reliability and scalability (over nodes) along with end-to-end resources, dynamic admission control and scalability (over network) for multicast applications.

As has been mentioned, multicast applications have varying performance requirements; therefore it is necessary to design a framework that guarantees divers quality of service. However the existing best effort network cannot provide the guaranteed service level required by multicast application. To overcome this problem, two solutions have already been proposed:

- Providing reliably and scalability during the connection;
- Providing end-to-end resources, dynamic admission control and scalability over a network for multicast applications.

The first solution proposed the tree-based functionality approach in the multicast transport protocol, to provide reliability and scalability between a sender and a group of receivers. The tree-based approach is characterized by dividing the receiver set into groups, distributing retransmission responsibility over an acknowledgement tree (ACK tree) structure built from the set of groups, with the source as the root of the tree. The ACK tree structure prevents receivers from directly contacting the source, in order to maintain scalability with a large receiver set. The ACK tree consists of receivers and the source (sender) organized into local groups, with each group having a group leader in charge of retransmission within the local group. However this solution only provides reliability and scalability for multicast applications, regardless of any quality service level required over the network environment.

The second solution has proposed end-to-end quality of service over the network environment using interoperation of Integrated services (IntServ) and Differentiated services (DiffServ) principles. Both QoS architectures, have their own advantages and disadvantages. IntServ architecture is defined as a set of extensions to the traditional best

effort model of the Internet with the goal of allowing end-to-end QoS to be provided to multicast applications. IntServ assumes that the guarantees for the delivery of multicast application can be achieved only by reserving resources from the network nodes. It only provides dynamic admission control, end-to-end predictable services and an efficient use of network resources. However, the processing overhead in the high-speed networks will be unacceptable.

On the other hand, Differentiated Services (DiffServ) architecture has benefited from the experiences of the Integrated Services architecture. The main concern of DiffServ is scalability. Scalability is achieved by aggregating the traffic classification state. However the lack of dynamic admission control mechanism and difficulties in resource allocation can make the DiffServ quite unpredictable.

With the interoperation of both architectures it might be possible to build a scalable system, which would provide predictable services. This framework has to be supported by multicast transport protocol to provide reliability and scalability over the nodes.

This research has proposed a framework to provide reliability and scalability on nodes (tree-functionality) along with end-to-end resources, dynamic admission control and scalability over the network (interoperation of IntServ and DiffServ) for multicast applications.

Enhanced communication transport protocol, (ECTP), transport protocol was chosen for this research for following reasons:

- ECTP transport protocol is a multicast transport protocol with tree-based functionality to support multicast applications. Tree-based functionality provides reliability and scalability for all multicast applications.
- ECTP transport protocol is able to provide QoS management functionality established by Integrated or/and Differentiated services to support multicast applications. QoS management functionality provides end-to-end resource

reservation, dynamic admission control and scalability over the network for multicast applications.

This research is focused on the further enhancement and implementation of the ECTP transport protocol, QoS management specification. It describes QoS management functions (negotiation, monitoring and maintenance) for the stable management of QoS for users in a simplex connection.

QoS management enables ECTP transport protocol to negotiate QoS management functionality provided by IntServ or/and DiffServ principles in networks. Therefore, the ECTP transport protocol is able to provide end-to-end resources, dynamic admission control and scalability on networks.

This research has used "*Optimized Network engineering tool*" (OPNET) to implement and investigate the ECTP specifications. OPNET simulator provides a comprehensive development environment for modeling and performance of communications networks.

ECTP specifications have been designed and implemented in the OPNET simulator (version 8.1). It enables OPNET to investigate and compare the behavior and performance of ECTP specifications with the UDP transport protocol (an existing transport protocol in OPNET) over voice application.

The investigation consists of three case studies. The simulation results have proved that the ECTP transport protocol with tree-based functionality and QoS management provided by IntServ and DiffServ, produced the best traffic delay for voice application.

This chapter presents an overview of this thesis, highlights its logical structure and discusses the future work. Figure (8.1) outlines the main objectives of this research.

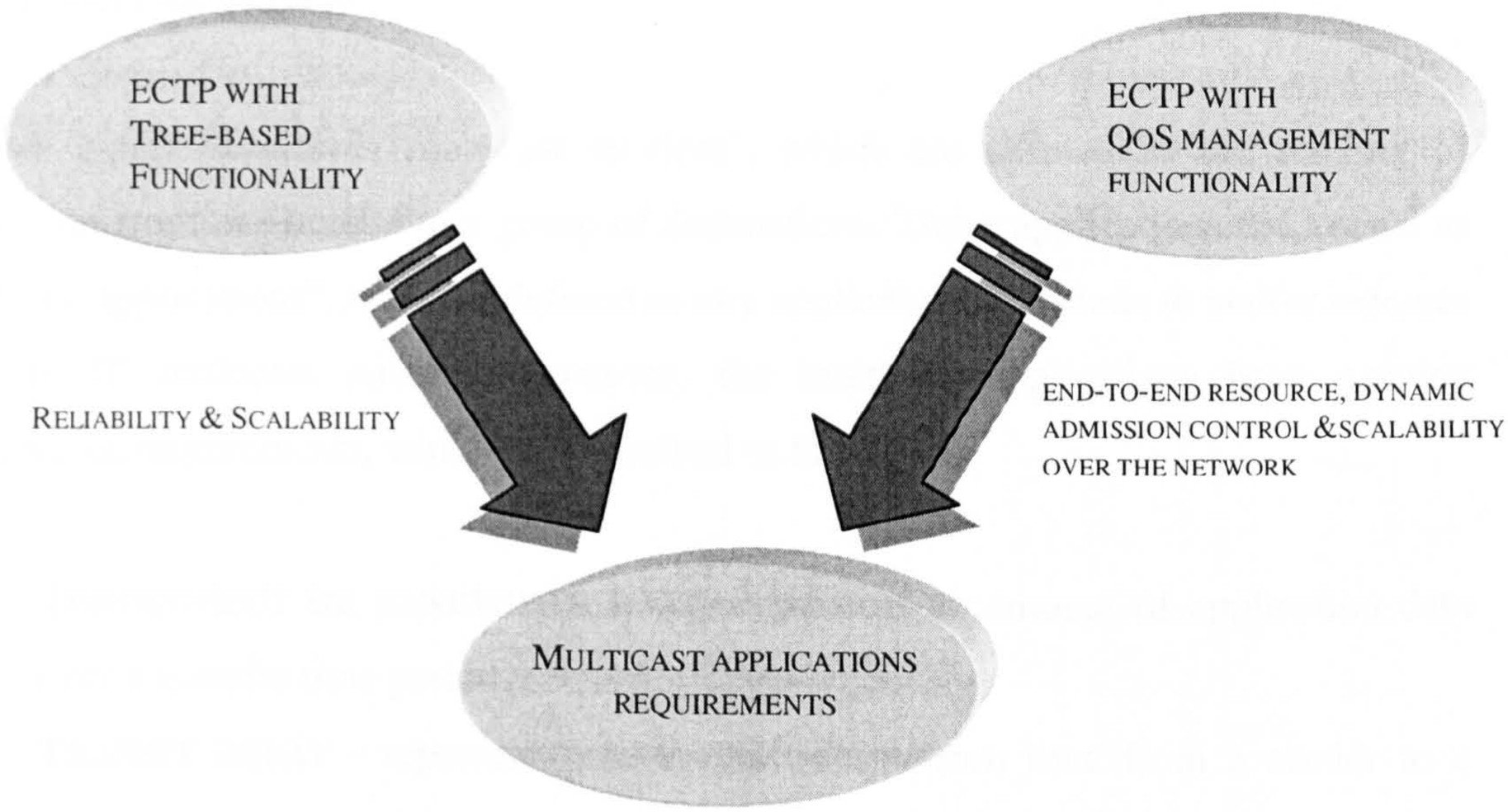


Figure (8.1): Thesis outline

8.1 LITERATURE REVIEW

Chapter 2 has discussed "multicast services", which are defined as the sending of application from one source to a group of destinations. These applications are known as "multicast applications", which is defined as any application that sends to and/or receives from an IP multicast address. However, the multicast applications have varying performance requirements, which are described as follows:

- **THROUGHPUT OR BANDWIDTH RATE**- represents an amount of application data over a specific time period.
- **TRANSIT DELAY** - represents end-to-end transmission time from a sender to a receiver
- **TRANSIT DELAY JITTER** - represents the variation of network delay
- **DATA LOSS**- represents the ratio of the amount of lost data to the amount of transmitted data.

These requirements constrain the design of the multicast transport protocol needed to transmit them over network. Therefore to overcome these problems, networks need a multicast transport protocol to provide scalability and reliability for multicast applications. Four approaches have been defined to enable scalability and reliability into the multicast transport protocol. They are described as follow:

- **SENDER-INITIATED APPROACH**- requires the sender to maintain a list (called ACK list) for each packet, of the receivers from which it has received a positive acknowledgement (ACK).
- **RECEIVER-INITIATED APPROACH** - places the responsibility for ensuring reliable packet delivery on the receiver. The sender continues to transmit new data packets until it receives a negative acknowledgment (NAK) from a receiver.
- **TREE-BASED APPROACH**- divides the receiver set into groups, distributing retransmission responsibility over an acknowledgement tree structure built from the set of groups, with the source as the root of the tree.

- **RING-BASED APPROACH-** supports applications that require total ordering of transmissions at all receivers. It has only one token site responsible for ACKing packets back to the source.

It has been mentioned that the experimental results have shown that the tree-based approach is the best answer to scalability and reliability. Enhanced communication transport protocol, ECTP, is a tree-based multicast transport protocol designed to support Internet multicast applications running over IP multicast capable networks. ECTP operates over IPv4/IPv6 networks that have the IP multicast forwarding capability. ECTP specifications have been defined by ITU.T/ISO in two parts: (i) control and (ii) QoS management for simplex connections.

ECTP-Control specification has already been completed, which specifies the protocol operations for connection establishment, reliability and scalability using the tree-based approach. This specification is presented in *Chapter 2*.

ECTP-QoS management specification describes QoS management functions for the stable management of QoS for users in a simplex connection. This research proposes to design of a framework to enable the ECTP transport protocol with the QoS management functionality.

Before discussing ECTP QoS management specification, *Chapter 3* presented an overview of QoS management operations, parameters and services. QoS management is defined as a concrete realization of required levels of QoS in a real system, which is described in three components: negotiation, maintenance and monitoring. QoS services are provided by two services: Integrated services (IntServ) and Differentiated services (DiffServ). IntServ architecture is designed to provide end-to-end resources and dynamic admission control over network. DiffServ architecture is designed to provide scalability over the network. The proposed interoperation of IntServ and DiffServ by IETF can achieve end-to-end resource, dynamic admission control and scalability over network providing end-to-end QoS for multicast applications.

8.2 CONTRIBUTION TO THE BODY OF KNOWLEDGE

The ECTP control specification has been completed by ITU.T/ISO, which specifies the protocol operations for connection establishment, reliability and scalability using a tree-based approach. However, the ECTP transport protocol cannot itself guarantee the QoS levels required by applications. *Chapter 4* proposed and defined new components to enable the ECTP transport protocol as a tree-based multicast protocol with QoS managements operations using IntServ and DiffServ principles.

During the ECTP connection lifetime, QoS negotiation is performed in the connection creations phase. After an ECTP connection is created, and if QoS management is enabled, the QoS monitoring is activated. Based on the aggregated QoS values, the sender takes the QoS maintenance actions to maintain the connection status at a desired QoS level.

Once all the new components have been defined in ECTP transport protocol, *Chapter 5* proposed two mapping models to enable ECTP transport protocol with end-to-end resource reservation, dynamic admission control and scalability using IntServ and/or DiffServ principles. They are described as follow:

- **MODEL (I)-** ***Enabling ECTP transport protocol to negotiate end-to-end resource reservation using standard RSVP signaling protocol (IntServ).***

  This model has proposed the mapping of ECTP QoS parameters target values onto Standard RSVP object traffic descriptors. As has been mentioned, the character of the ECTP transport protocol itself is unable to provide QoS management. Therefore an interface between ECTP and RSVP is required so that the ECTP QoS parameter target values would be informed to the RSVP processor. This mapping would enable the ECTP transport protocol to provide end-to-end connection for multicast applications by negotiating the QoS levels in the connections creation phase. QoS negotiation operation uses RSVP signaling to enable ECTP to negotiate the resource reservation required by applications to all

active receivers before transmitting packets. This model followed the IntServ principle, which only provides end-to-end resources and dynamic admission control. Therefore Model (II) has been proposed to provide scalability as well as end-to-end resources and dynamic admission control.

- **MODEL (II)-** *Enabling ECTP transport protocol to negotiate end-to-end resource reservation using both the standard and aggregated RSVP signaling protocol (interoperation between IntServ and DiffServ).*

This model proposed the mapping of ECTP QoS parameter target values into standard and aggregated RSVP object traffic descriptors. This mapping should enable the ECTP transport protocol to provide end-to-end connection for multicast applications by negotiating the QoS level in the connection phase. Therefore, ECTP QoS parameter target values are mapped into the standard and aggregated RSVP resource reservation (object traffic descriptors) to provide end-to-end resources and dynamic admission control as well as scalability over networks.

8.2.1 IMPLEMENTATION AND SIMULATION RESULTS

Once the architectures of two mapping models are proposed and defined, *chapter 6* explained the implementation of ECTP transport protocol specifications: control and QoS management (model (I) and (II)) into OPNET).

The implementations have followed the OPNET modeling methodology to define each process model. Each process model is expressed in Proto-C, a language combining graphical state-transition diagrams (STDs), embedded C/C++ language data items and statements, and library of Kernel Procedures (KPs) that provides commonly needed functionality for modeling communications and information processing systems. Table (8.1) summarizes the implemented process models in OPNET simulator.

IMPLEMENTED PROCESS MODEL	MODELS
ECTP transport protocol	ectp_manager ectp_connection ectp_QoS_management rsvp aggrsvp
DS router process model	DS_ip DS_ps ip_icmp

Table (8.1): Implemented process models in OPNET simulator

Once the ECTP transport protocol specifications have been implemented and defined in OPNET, *Chapter 7* discussed three case studies to simulate and analyze the behavior and performance of the each of the ECTP specifications over the network model. Each case study has been assumed to use single path methodology network model and explicit traffic specification. The case studies are defined as follows:

- **CASE STUDY (I): ECTP WITHOUT RSVP**

  This investigates the behavior and performance of the ECTP transport protocol enabled without RSVP resource reservation using voice application.

- **CASE STUDY (II): ECTP WITH STANDARD RSVP**

  This investigates the behavior and performance of the ECTP transport protocol enabled by standard RSVP resource reservation using voice application.

- **CASE STUDY (III): ECTP WITH STANDARD AND AGGREGATED RSVP**

  This investigates the behavior and performance of the ECTP transport protocol enabled by the standard and aggregated RSVP resource reservation using voice application.

All the case studies have been compared with the exiting transport protocol, UDP using traffic delay parameter for voice application. The simulation results are analyzed for all three case studies have showed the following results:

- The performances of the ECTP transport protocol without RSVP resource reservation on traffic delay parameter for voice application have shown that ECTP can provide improved QoS compared to the UDP transport protocol without RSVP resource reservation. The main reason is that with the ECTP tree-based functionality, it can provide better reliability and scalability for voice application.

- The performances of the ECTP transport protocol with standard RSVP resources reservation (model (I)) on the traffic delay parameter over voice applications have shown that ECTP can provide better QoS compared to the UDP transport protocol with or without RSVP resource reservation. The main reason is that with the ECTP tree-based functionality and QoS management (standard RSVP), it can provide better reliability as well as end-to-end resources and dynamic admission control for voice application.

- The performances of ECTP transport protocol with standard and aggregated RSVP resources reservation (model (II)) on traffic delay parameter over voice application have shown that ECTP can provide improved QoS compared to the UDP transport protocol with or without RSVP resource reservation. The main reason is that the ECTP tree-based functionality and QoS management (standard RSVP) can provide better reliability as well as end-to-end resources, dynamic admission control and scalability over networks for voice application.

- ECTP transport protocol with standard and aggregated RSVP resource reservation has the best performance compared to ECTP with or without standard RSVP resource reservation on traffic delay parameters to deliver voice application over the network. The main reason is that ECTP transport protocol with standard and aggregated RSVP resource reservation provides

reliability, dynamic admission control, efficient network resources and scalability over networks.

8.3 FUTURE WORK

This work can be applied on the further enhancement of ECTP transport protocol specifications for N-plex multicast transport connection.

As it mentioned in Chapter 2, the ECTP control specification employs the tree-based control mechanism to achieve high scalability as well as reliability by adopting hierarchical tree-based control architecture. However, the ECTP tree-based control mechanism has been defined only for a simplex multicast transport connection, in which there are only one sender and a group of receivers. They are known as the root and local groups respectively. Figure (8.2) illustrates the simplex multicast transport connection.

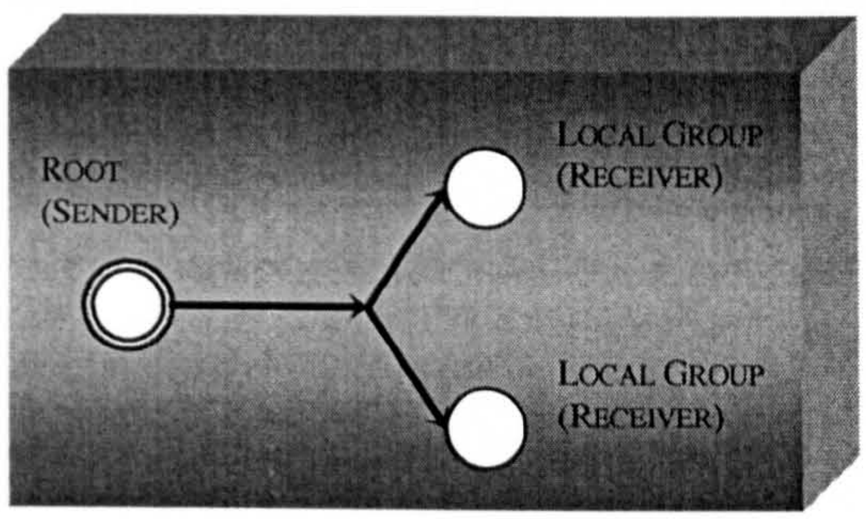


Figure (8.2): Simplex multicast transport connection

The future work could focus on the further enhancement of the ECTP tree-based control specification for an N-plex multicast transport connection. An N-plex multicast transport connection is defined as a multicast connection, which all the nodes (senders and receivers) can act as either root or local groups. Figure (8.3) illustrates the N-plex multicast transport connection.

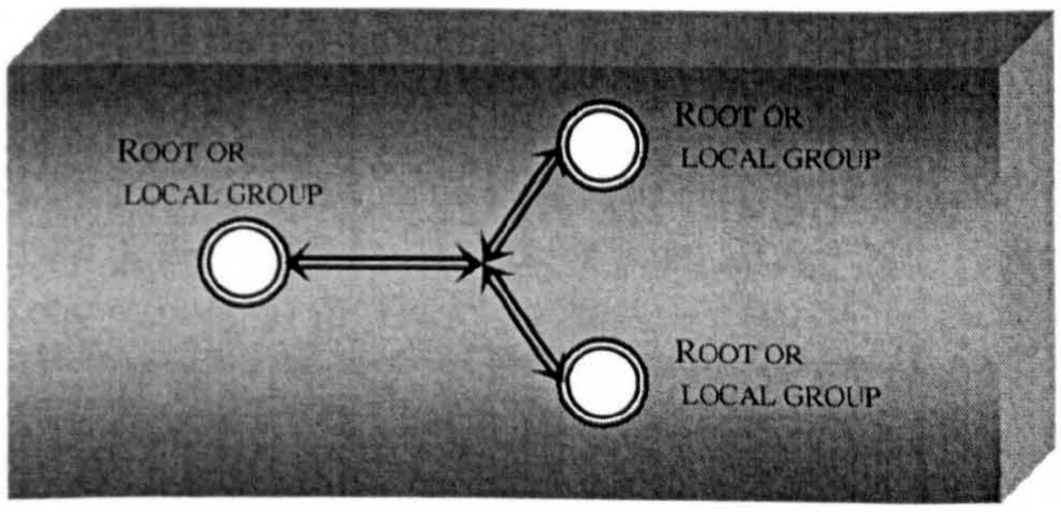


Figure (8.3): N-plex multicast transport connection

Any root in N-plex multicast connection can be assigned as the connection owner. That root is at the heart of multicast group communications. It is responsible for overall connection management by governing connection creation, data transmission and connection terminations. ECTP can employ shared tree-based control mechanisms for connection creation and termination, which are configured during connection creation. The control tree can define a parent-child relationship between any pair of tree nodes.

In an N-plex multicast transport connection, data transmission in connection management can be implemented using as a token-based approach, in which the connection owner manages the token(s) to transmit data.

According to the definition of ECTP tree-based control for an N-plex multicast transport connections, it can be employed the QoS management function to provide end-to-end QoS.

REFERENCES

A

[AB+04] Adamson B., Bormann C., Handly M., Macker J., "Negative-acknowledgment (NACK)-Oriented Reliable Multicast (NORM) Protocol", RFC 3940, Network working group, IETF, Novermber 2004

[Almeroth99] Almeroth K., " Managing IP multicast traffic: A first look at the issues, tools and challenges", white paper, IP multicast Initiative, January 1999

[Almquist92] Almquist P., "Type of Service in the Internet Protocol Suite", RFC 1349, IETF, July 1992

[ATMFORUM90] "ATM Traffic Management Specification Version 4.0", ATM Forum, April 1996. ANSI T1S1, March 1990

[AS01] Auerbuch B., Shavitt Y., " Topology of aggregating traffic", IEEE/ACM Transaction on Networking, Vol. 9, No.1, February 2001

B

[BB+98] Blake S.., Black D., Carlson M, Davies E., Wang Z. , Weiss W., "An Architecture for Differentiated Services", RFC 2475, DiffServ working group, IETF, December 1998

[BC+94] Braden R., Clark D., Shenker S. "Integrated Services in the Internet Architecture: an Overview", RFC 1633, IntServ working group, IETF, Jun 1994

[BD+01] Beccheti L., Delli F., Inzerilli T., Mahonen P., Munoz L., " QoS support for an ALL-IP system beyond 3G.", IEEE communication magazine, Vol. 39, No.8, pp.64-73, August 2001

[Bernet00] Bernet Y., " The complementary roles of RSVP & Differentiated services in the full-service QoS network", IEEE communication magazine, Vol. 38, No.2, pp.154-165, February 2000

[BF+00] Bernet Y , Ford P., Yavatkar R., Baker F., Zhang L., Speer M., Braden R., Davie B., Wroclawski J., Felstaine E., "A Framework for Integrated Services Operation over Diffserv Networks", RFC 2998, DiffServ working group, IETF, November 2000

[BI01] Baker, F. Iturralde C., Faucheur F. Le, Davie B., "Aggregation of RSVP for IPv4 and IPv6 Reservations" RFC 3175, IP multicast working group, IETF, September 2001

[BO+97] Berger L, O'Malley T., "RSVP Extensions for IPSEC Data Flows", RFC 2207, IETF, September 1997

[Bufrod94] Buford JFK., "Multimedia Systems", Addison-Wesley, 1994;

[BH98] Branden R., Hoffman D., "RAPT- an RSVP Application Programming Interface Version 5", Internet Draft, August 1998

[BV98] Berson S. and Vincent S., "Aggregation of Internet Integrated Services State." In 6th IEEE/IFIP International Workshop on Quality of Service (IWQoS'98), Napa, California, USA, May 1998.

[BZ97] Braden R., Zhang L., "Resource ReSerVation Protocol (RSVP) -- Version 1 Message Processing Rules", RFC 2209, IETF, September 1997

[BZ+97] Braden R., Zhang L., Berson S., Herzog S., Jamin S. "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification" RFC 2205, IntServ working group, IETF, September 1997

C

[CE98] Choe J, Eunsook K.,, "An Efficient Restricted Retransmission Mechanism for Error Recovery of a Local Group", JCCI'98, April 1998

[Clark88] Clark, D., "The Design Philosophy of the DARPA Internet Protocols", ACM SIGCOMM '88, August 1988.

[CM84] Chang J-M, Maxemchuk NF. "Reliable broadcast protocols" ACM trans computing system, pp. 251-273, February 1984

[CN97] Chiueh T., Neogi A., " Performance Analysis of an RSVP-Capable Router" , IEEE Real Time Technology and Applications Symposium, 1998

[Comer91] Comer, Douglas E, "Internetworking with TCP/IP v.1 principles, protocols and architecture", Second Edition, 1991

[CP+00] Choe J, Park J, Kang S.G, Koh S.J, Eunsook K., " A Design mechanism of well-balanced ACK Trees for Tree-based Reliable Multicast Protocols", The 6th Asia-Pacific Conference on Communications (APCC2000), October 2000

[CS92] Clark, D., Shenker, S. and Zhang, L., "Supporting Real- Time Applications in an Integrated Services Packet Network: Architecture and Mechanism," in Proc. SIGCOMM'92, September 1992

[CS+92] Clark D., Shenker S., Zhang L., " Supporting Real-time applications in Integrated packet services network", SIGCOM'92, Baltimore, MD, August 1992

[CW+03] Chiu D., Whetten B., Kadonsky M., Koh S., Takale G, "Reliable Multicast Transport Building Block: Tree based ACK (TRACK) Mechanisms", Internet Darft, RMT working group, IETF, December 2003

D

[Deering89] Deering S. " Host extension for IP multicasting" RFC 1112. Working group IP multicast, IETF, 1989

[Detti99] Detti et al. " Supporting RSVP in Differentiated service Domain: an architectural framework and scalability analysis" ICC '99, Vancouver June1999, pp. 204-210

[Dobrowski98] Dobrowski G.H., "The ATM forum: Developing implementation agreements", IEEE communication magazine, September 1998, p.121-125

[DP+99] Doverspike R.D., Phillips S., Westbrook J.R, " Future transport network Architecture", IEEE communication magazine, Vol. 37, No. 8, pp. 96-100, August 1999

[Durand01] Durand A., "Deploying IPv6", IEE Internet computing magazine, January/ February 2001, Vol. 5., No.1

E

[EA+00] Engell A.N., Azcorra A., Bertelsen E., Carapinha J., Dittmann L., Kjaergarrd K.J., Mckay I., Maliszewski J., Papir Z., " Applying QoS control through Integration of IP", IEEE communication magazine, Vol. 38, No.7, pp. 130-136, July 2000

[EC00] Eunsook K., Choe J., "A Mechanism for Constructing Efficient Transport Layer Trees for Reliable Multicast Service", KISS 2000, April 2000

[EH+00] Eichler G., Hussmann H., Mamais G., Venicris I., Pehofer C., Salsano S., " Implementing Integrated and Differentiated services for Internet: A practical approach." IEEE communication magazine, Vol. 38, No.1, January 2000

[EK+01] Eunsook K., Kang S.G., Choe J., Koh S.J, Park J, "An ACK-Tree Creation Mechanism using Top-down scheme for Tree-based Reliable Multicast Transport Protocols", International Conference on Internet Computing (IC2001), Jun. 2001

[Estrin98] Estrin D. " Multicast: enabler and challenges", Cattech Earthlink seminar series, 1998

[Etal99] Et al J.V. "An Expedited Forwarding PHB", RFC 2598, DiffServ working group, IETF, June 1999.

F

[Fenner97] Fenner W. , "Internet Group Management Protocol, Version 2", RFC - 2236, IP multicast working group, IETF, November 1997

[FH98] Ferguson S., Huston G., " Quality of Service: Delivering QoS on the Internet and in corporate networks", John Wiley & Sons, 1998, p.66

H

[HC01] Holbrook H., Cain B. "Source-specific multicast for IP", working group IP multicast, 52nd IETF meeting, 2001

[HF+00] Hondey M., Floyd S., Whetten B., Kermode R., Vicisano L., Luby M. " The reliable multicast design space for bulk data transfer", RFC 2887, working group RMT, IETF, August 2000

[HJ98] Handley M., Jacobson V. , "SDP: Session Description Protocol", RFC - 2327, MMUSIC working group, IETF, April 1998

[HNRS99] Hofman, Nonnenmacher, Rosenberg, Schulzrinne "A taxonomy of feedback for multicast", 50th IETF meeting, 1999

[HP+00]Handley M., Perkins C., Whelan E., "Session Announcement Protocol", RFC - 2974, MMUSIC working group, October 2000

I

[IETF] http:// www.IETF.org

[IETF-DiffServ] DiffServ working group, http:// www.ietf.org/html.charters/DiffServ-charter.html

[IETF-IntServ] IntServ working group, http:// www.ietf.org/html.charters/inteserv-charter.html

[IK+00] Iida K., Kawahara K., Takine T., Oie Y., " Performance evaluation of architecture for end-to-end Qulaiy of service provisioning.", IEEE communications magazine, Vol.39, No.4, pp.76-81, April 2000

[ISO] International Standard organisation, http:// www.ISO.org

[ISO/IEC98] "Local and metropolitan area networks - Common specifications - Part 5: Token Ring Access Method and Physical Layer Specifications", ISO/IEC 8802-5 Information technology- Telecommunications and information exchange between systems, 1998

[ISO/IEC00] Local and metropolitan area networks - Common specifications - Part 3: Media Access Control (MAC) bridges, (current draft available as IEEE P802.1D/D15). ISO/IEC Final CD 15802-3 Information technology - Tele- communications and information exchange between systems, 2000

[ITU.T] International telecommunication Union, http:// www.ITU.org

J

[JH+99] Johson D., Hayman N., Veitch P., " The evolution of a reliable transport network", IEEE communication magazine, Vol. 37, No. 8, pp. 54-57, August 1999

[JN98] Jacobson, V., Nicholas K., " The Internet Protocol", Cisco Internet protocol Journal, Jan 1989

[JS+92] Jamin S., Shenker S., Zhang L, Clark D.D. ,"An Admission Control Algorithm for Predictive Real-Time Service", In Proceeding of the 3rd International workshop on network & operating systems support for digital audio and video, NOSSDAV'92, La Jolla, CA, Nov. 1992, pp. 73-91

K

[KC+99] Kim Y., Choe J.,Park J,, , Kang S.G, Koh S.J, Eunsook K,Non-core based shared tree architecture for IP multicasting", IEE Electronics Letters, Vol. 35, No. 11, pp. 872 - 873, May, 1999

[KC+01] Kang S.G., Choe J., Eunsook K., Koh S.J, Park J., "A Design and Analysis of Top-down based ACK-Trees", Proc. 9th IEEE International Conference On Networks, October 2001

[KCP+02] Kim Y., Choe J.,Park J,, , Kang S.G, Koh S.J, Eunsook K, " A router assisting control tree configuration mechanism for reliable multicast", LNCS 2343 (Information Networking: Wired Communications and Management), pp. 84 - 93, June 2002.

[KE+01]Koh S.J, Eunsook K., Park J, Kang S " Configuration of ACK trees for multicast transport protocols.", ETRI Journal, Vol. 23, No. 3, Septemeber2001

[Kilkki99] Kilkki K., " Differentiated services for Internet", Macmillan Technical publishing, 1999

[KK+02] Koh S.J., KIM D.Y., Kang, S.G., " Enhanced communication transport protocol for multicast transport", LNCS 2343 (Information Networking: Wired Communications and Management), pp. 64 - 74, June 2002.

[KKP+02] Koh S. J., Kim E., Park J., Kang S. G., " Framework of QoS management for multicast communications", ICACT 2002, pp. 46 - 50, February, Korea, 2002

[Koh01] Koh S.J. (Project Editor) "ECTP Control Specification", ITU-T Recommendation X.606/ISO/IEC FDIS 14476-1 (ECTP-1), 2001,

[Koh02] Koh S.J. (Project Editor) "ECTP QoS management Specification", ITU-T Recommendation X.606/ISO/IEC FDIS 14476-2 (ECTP-2), 2002,

[KP+00] Kim Y., Park J,, , Kang S.G, Koh S.J, Eunsook K "A Framework of Connection Management for Multicast Transport", Proceeding of the APCC2000, South Korea, October 2000

[KPC+00] Kim Y., Park J, Choe J., ,, , Kang S.G, Koh S.J, Eunsook K, "Minimising cast and delays in shared multicast transport", ETRI Journal, Vol. 22, No. 1, pp.30 - 37, March 2000.

L

[LB01] Legout A., Biersack E.W., "Bandwidth-Allocation Policies for unicast and multicast flows", IEEE/ACM Transactions on Networking, Vol.9, No.4, August2001

[LG98] Levine B. N., Garcia-Luna-Aceves J.J., "A comparison of reliable multicast protocols", Multimedia Systems, Springer-Verlag, 1998

[LL+96] Levine BN., Lavo D., Garcia-luna-Aceves JJ. " The case for reliable concurrent multicasting using shared ACK trees" Proc. ACM multimedia, pp 365-376, Nov 1996

[LP96] Lin J., Paul S., " A reliable multicast transport protocol", IEEE Infocom, pp.1414-1425, September 1996

[LVG96] Levine B. N., Garcia-Luna-Aceves J.J., "Comparison of Known Classes of Reliable Multicast Protocols." IEEE computer society press, pp. 112-121, November 1996

[LV+02] Luby M, Vicisono L, Gemmell J., Rizzo L., Handly M., Crowcorft J., "Forward Error Correction (FEC) Building Block", RFC 3452, Network working group, IETF, July 2002

M

[Maufer98] Maufer T.A., "Deploying IP multicast in Enterprise", Prentice-Hall, 1998

[MB+97] Mankin A., Baker F., Braden B., Bradner S., O`Dell M., Romanow A., Weinrib A., Zhang L. , "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment", RFC 2209, IETF, September 1997

[MB+99] Manchester J., Bonenfant P., Newton C., " The evolution of transport network survivability", IEEE communication magazine, Vol. 37, No. 8, pp.44-51, August 1999

[MCDysan00] McDysan D. "QoS & Traffic management in IP & ATM networks", McGrawhill, 2000

[Miller99] Miller C.K., "Multicast networking and Applications", First Edition, 1999

[Mills89] Mills D. L., "Network Time Protocol (Version 2) Specification and Implementation", RFC 1119, IETF, September 1989

[Mir01] Mir N.F., "A survey of data multicast techniques, architectures, and algorithms", IEEE Communications Magazine, Vol.39, No.9, pp.164-169 September 2001;

[Moy97] Moy J.T. "Anatomy of An Internet Routing Protocol", Addison-Wesley, 1997

[MR+98] Mankin A., Romanou A., Bradner S., Paxson V. "IETF criteria for evaluating reliable multicast transport and application protocols", RFC 2357, working group RMT, IETF, June 1998

N

[NB+98] Nichols K., Blake S., Baker F., Black D., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DiffServ working group, IETF, December 1998

[NJ+99] Nichols D., Jacobson V., Zhang L., "A Two-bit Differentiated Services Architecture for the Internet", RFC 2638, DiffServ working group, IETF, July 1999

O

[Obraczka98] Obraczka, K. "Multicast Transport Protocols: A Survey and Taxonomy ", IEEE Communication Magazine, January 1998

[OPNET_Concept] , "OPNET modelling concept", OPNET simulator tool, version 8.1

[OPNET_Kernel], "OPNET Simulation Kernel-Introduction", OPNET simulator tool, version 8.1

[OPNET_Methodology], "OPNET Modelling Methodology", OPNET simulator tool, version 8.1

[OPNET_Modeling], “Modeling Network Traffic”, OPNET simulator tool, version 8.1

[OPNET_ Process] "Modelling concepts- Process model operation", OPNET simulator tool, version 8.1

[OPNET_STD], " Modelling concepts-States transition diagrams", OPNET simulator tool, version 8.1

[OPNET_Traffic], "Representing Network Traffic", OPNET simulator tool,

[OPNET_Topology], "Building Network Topologies", OPNET simulator tool, version 8.1

[OS98] Orezessek M., Sommer P., "ATM- MPEG-2 Integrating Digital video into broadband networks", Prentice Hall, 1998

P

[PA+01] Pekowsky S., Andorfer A., Digital P., "Multimedia Data Broadcasting Strategies", IEEE communication magazine, Vol.39, No.4, pp.138-145, April 2001

[PC+01] Park J, Choe J, Kang S.G, Koh S.J, Eunsook K. "A Control Tree Creation Mechanism for Reliable Multicast Transport Protocols", The 3rd International Conference on Advanced Communication Technology (ICACT2001), February 2001

[PE+01] Park J, Eunsook K., Kang S.G., Choe J., Koh S.J "Sender-initiated Configuration of an ACK-Tree for Reliable Multicast Transport", 11th International Packet Video Workshop (PV2001), April 2001

[Pessi97] Pessi P., "RSVP and the Internet Integrated services"- Internal report, Nockia Research Centre- Communication Systems Laboratory, August 1997

[PK+00] Park J,, , Kang S.G, Koh S.J, Choe J, Eunsook K., "A loop-free guaranteed ACK-Tree Connection Mechanism", KICS2000, November. 2000

[PK02] Park J., Koh S.J., " A new delivery scheme for 1-to-N multicast applications", *LNCS 2343 (Information Networking: Wired Communications and Management)*, pp. 109 - 118, June 2002.

[PMB99] Pullen J., Mytak M., Bouwens C. " Limitation of Internet protocol suite for distribution simulation in large multicast environment" RFC 2502, IETF, 1999

[Postel80] Postel J. "UDP: User Datagram Protocol", RFC 768, Transport Working group, IETF, August 1980

[Postel81] Postel J., "TCP: Transmission Control Protocol", RFC 793, Transport working group, IETF, September 1981

[PostelJ81] Postel J. "Internet control message protocol-ICMP", RFC 792, IETF, September 1981

[Postel82] Postel J, B., "Simple Mail Transfer Protocol", RFC 821, IETF, August 1982

[PT+94] Pingle S., Towsley D., Kurose J.F., " A comparison of sender-initiated and receiver-initiated reliable multicast protocols", In Performance Evaluation Review, May 1994, University of Massachusetts

Q

[QA01] Quinn B., Almeroth K., "IP multicast: challenges and solutions", RFC 3170, working group IP multicast, IETF 2001

[QK01] Qiu Y., Knightly W., " Measurement-based Admission control with aggregated traffic.", IEEE/ACM Transaction on Networking, Vol. 9, No.9, April 2001

R

[RF95] Romanow, A., and Floyd, S., "Dynamics of TCP Traffic over ATM Networks", IEEE JSAC, V. 13 N. 4, May 1995, p. 633-641

[RJ87] Ramakrishnan S., Jain BN., " A negative acknowledgement with periodic polling protocol for multicast", IEEE information, pp 502-511, 1987

[Rutkowski99] Rutkowski A.M., " understanding next generation Internet", IEEE//ACM transactions on Networking, Vol.8, No.3, March 1999

S

[Stevens94] Stevens W.R. "TCP/IP Illustrated, Volume 1" , W.R. Stevens. 1994

[Shenker95] Shenker, S., " Fundamental Design Issues for the Future Internet", IEEE Journal on Selected Areas in Communications, Vol.7, No. 6, September 1995,

[SP+97] Shenker S., Partridge C., Guerin R., "Specification of Guaranteed Quality of Service", RFC 2212, Intserv working group, IETF, September 1997

[Stallings98] Stallings, W., "high-speed networks, TCP/IP & ATM design principals", Prentice-Hall, 1998

[Stiller98] Stiller, B. " Quality of service issues in Networking environments", Technical report No. 380, University of Cambridge, Computer laboratory,1998

[Stuart99] Stuart J.D., "QoS protocols and architectures", White paper, QoS forum, July1999

T

[Topolcic90] Topolcic, C., "Experimental Internet Stream Protocol: Version 2 (ST-II)", RFC 1190, IETF, October 1990

[TW97] Tsaoussidis V. and Wei S., " QoS Management at the Transport Layer", IEEE communications magazine, Vol.5, No.9, pp. 113-124, March, 1999

V

[VK99] Veeragaghavan M., Karbol M., “Interworking connectionless and connection-oriented networks” IEEE communications magazine, Vol.37, No. 12, pp.130-135, December 1999

W

[White97] White P. P., "RSVP and integrated services in the Internet: A tutorial," IEEE Communication Magazine, Vol.4, No.8, pp. 100-106, May 1997

[Wroclawski97] Wroclawski J., "Specification of the Controlled-Load Network Element Service", RFC 2211, Intserv working group, , IETF, September 1997

X

[XL99] Xipeng X., Lioinel M., " Internet QoS: A big picture", IEEE network magazine,Vol. 6, No.8, March 1999, pp. 8-18

Z

[ZD+93] Zhang L., Deering S., Estrin D., "RSVP: A new resource reservation protocol", IEEE Network, September 1993

APPENDICES

APPENDIX (A)

TYPE OF MULTICAST APPLICATION

TYPES OF MULTICAST APPLICATION

- **SCHEDULED AUDIO/VIDEO (A/V) DISTRIBUTION**

  Lectures, presentations, meetings, or any other type of scheduled event whose multimedia coverage could benefit an audience (i.e. television and radio "broadcasts"). One or more constant-bit- rate (CBR) datastreams and relatively high-bandwidth demands characterise these applications. When more than one datastream is present--as with an audio/video combination--the two are synchronised and one typically has a higher priority than the other(s). For example, in an a/v combination it is more important to ensure an intelligible audio stream, than perfect video.

- **PUSH MEDIA**

  News headlines, weather updates, sports scores, or other types of non-essential dynamic information. "Drip-feed", relatively low-bandwidth data characterise these applications.

- **FILE DISTRIBUTION AND CACHING**

  Web site content, executable binaries and other file-based updates sent to distribute end-user or replication/caching sites.

- **ANNOUNCEMENTS**

  Network time, multicast session schedules, random numbers, keys, configuration updates, (scoped) network locality beacons, or other types of information that are commonly useful. Their bandwidth demands can vary, but generally they are very low bandwidth.

- **MONITORING**

  Stock prices, Sensor equipment (seismic activity, telemetry, meteorological or oceanic readings), security systems, manufacturing or other types of real-time information. Bandwidth demands vary with sample frequency and resolution, and may be either constant-bit-rate or bursty (if event- driven).

- **MULTIMEDIA CONFERENCING**

  Audio/Video and whiteboard comprise the classic conference application. Having multiple datastreams with different priorities characterises this type of application. Co-ordination issues--such as determining who gets to talk when--complicate their development and usability. There are common heuristics and "rules of play", but no standards exist for managing conference group dynamics.

- **SYNCHRONISED RESOURCES**

  Shared distributed databases of any type (schedules, directories, as well as traditional Information System databases).

- **CONCURRENT PROCESSING**

  Distributed parallel processing.

- **COLLABORATION**

  Shared document editing.

- **DISTANCE LEARNING**

  This is a one-to-many a/v distribution application with "upstream" capability that allows receivers to question the speaker(s).

- **CHAT GROUPS**

  These are like text-based conferences, but may also provide simulated representations ("avatars") for each "speaker" in simulated environments.

- **DISTRIBUTED INTERACTIVE SIMULATIONS (DIS)**

  Each object in a simulation multicasts descriptive information (e.g., telemetry) so all other objects can render the object, and interact as necessary. The bandwidth demands for these can be tremendous, as the number of objects and the resolution of descriptive information increases.

- **MULTI-PLAYER GAMES**

  Many multi-player games are simply distributed interactive simulations, and may include chat group capabilities. Bandwidth usage can vary widely, although today's first-generation multi-player games attempt to minimise bandwidth usage to increase the target audience (many of whom still use dial-up modems).

- **JAM SESSIONS**

  Shared encoded audio (e.g., music). The bandwidth demands vary based on the encoding technique, sample rate, sample resolution, number of channels, etc.

- **RESOURCE DISCOVERY**

  Service Location, for example, leverages IP Multicast to enable something like a "host anycasting service" capability (anycast): A multicast receiver to send a query to a group address, to elicit responses from the closest host so they can satisfy the request. The responses might also contain information that allows the receiver to determine the most appropriate (e.g., closest) service provider to use. In Table 1, this application is entry D4. It is also illustrated in Figure 2 by possibility number 3. Alternately, the response could be to a multicast rather than unicast address, although technically that would make it an MtoM application type (this is how the Service Location Protocol (SLP) operates, when a user agent attempts to locate a directory agent).

- **DATA COLLECTION**

  This is the converse of a one-to-many "monitoring" application described earlier. In this case there may be any number of distributed "sensors" that send data to a data collection host.
  The sensors might send updates in response to a request from the data collector, or send continuously at regular intervals, or send spontaneously when a pre-defined event occurs. Bandwidth demands can vary based on sample frequency and resolution.

- **AUCTIONS**

  The "auctioneer" starts the bidding by describing whatever it is for sale (product or service or whatever), and receivers send their bids privately or publicly (i.e., to a unicast or multicast address).

- **POLLING**

  The "pollster" sends out a question, and the "pollees" respond with answers.

- **JUKEBOX**

  Allows near-on-demand a/v playback. Receivers use an "out-of-band" protocol mechanism (via web, email, unicast or multicast requests, etc.) to send their playback request into a scheduling queue (JMJThe initial unicast request is the only difference between this type of application and a typical 1toM. If that initial request were sent to a multicast address, this would effectively be an MtoM application.

- **ACCOUNTING**

  This is basically data collection but is worth separating since it is such an important application. In some multicast applications it is imperative to know information about each receiver, possibly in real-time. But such information can be overwhelming (MRM).
  Current mechanisms, like RTCP (which is actually MtoM since it is multicast but could be made Mto1), use scaling techniques but trade-off information granularity. As a group grows the total amount of feedback is constant but each receiver sends less.

APPENDIX (B)

GROUP DESCRIPTION IN MULTICAST CONNECTION (ITU.601)

GROUP DESCRIPTION IN MULTICAST CONNECTION (ITU.601)

- Multicast connection operation may be achieved by five distinct phases, which are described as follow:

1. MULTICAST GROUP CREATION PHASE

This operation is to create a multicast-group. The multicast-group may be created by specifying the set of rules that defines the future enrolled-groups memberships. Before this operation, the multicast-group does not exist and this operation will be performed by the group management facility.

As soon as multicast-group is reported to the group manger with group membership criteria, a group-name is assigned to the group with appropriate group membership criteria.

Then the information about this multicast-group is to be known to the potential service users. Therefore, a multicast-group can be identified by its group name in the following actions by the potential service users.

These criteria may be defined by means of set of rules or by more arbitrary means such as a list. The group criteria may include group-application name, group-name(s), proposed group-characteristics, and so on, depending on the hierarchy that the group is located.

2. REGISTRATION/DE-REGISTRATION PHASE

After creation of multicast-group, a service user should make it known to the group manager that it intends to be a member of enrolled-group.

Registration operation can be presented by letting the group manger to know group names and addresses of multicast-group members. Therefore, registration-group is composed of members of multicast-group that have successfully performed the registration operations.

It may be possible for a certain service user, such as group-owner, or group-manager to determine the members of the registered group.

De-registration, the inverse operation of registration, is performed when a service user registration is deleted from the list of registered group members, or from the forwarding list in the intermediate systems i.e. router.

3. ENROLLMENT/DE-ENROLLMENT PHASE

Enrollment operation allows a service user already registered to become a member of an enrolled-group.

The enrolled-group is consisted of members that have successfully performed the registration and the enrolment operations.

After creation of registered-group, a group-address is assigned from the group manager, and is known to the registered-group members. An enrolled-group is actually in the position to be reached by means of at least one group-address or group-name.

A member of an enrolled group can also be reached by its individual address. Then registered-group members allow an intermediate system like a router to activate required transmission channel(s) and forward the data to itself.

The group enrollment operation accomplishes the whole set-up that is necessary to place an entity into the position to be instantiated as a member in an actual enrolled-group.

For example, in the Internet applications, they select desirable group-connection(s) in a proposed group-association, letting those selected group-address(s) be activated in a designated router, which is capable of multicast demon, using IGMP, and IP multicast routing protocols to forward the data to the end-system.

Even though the data is forwarded to the end system, it may not be received properly since the end system is not yet been activated.

De-enrollment, the reveres operation of enrollment, is performed when an n enrolled-service user is deleted from the list of enrolled-group members, or from the forwarding list in the intermediate system. Then the enrolled-group member permits an intermediate system like a router to release the transmission channel

(s), so that the enrolled-group member is in the position that can neither be reached through the group-address nor by the group-name.

4. ACTIVATION/ DE-ACTIVATION PHASE

The activation phase establishes the shared state of specific instant of multicast communication based on information established during the enrollment phase among the participants.

During the activation phase the specific QoS requirements would be accepted for data transfer must be made or modified if they were not made during the enrollment phase.

In this phase, group association (GA) and/or group connection is created. The member that initiates the creation of GA is called the initiator.

The de-activation operation is performed to terminate a GA. This may occur at any time and may or may not be abrupt. It may be initiated either by the service-user or the service-provider.

5. DATA TRANSFER PHASE

The data transfer phase involves the actual transfer of data among the group members who belong to the active-group. It may include functions termination the QoS, either in terms of error control or resource reservation.

- **ECTP ENROLLMENT PHASE BEFORE CREATION CONNECTION**

Before an ECTP connection is created, each potential group member (receivers) must be enrolled to the multicast group, which it is called an *enrolled group member*. Some or all of the enrolled group users will participate in the ECTP connection.

Before an enrolled group member joins the multicast connection, it must be attached to the network interface with the help of Internet group management protocol (IGMP) and IP multicasting routing protocols. These protocols ensure that the enrolled member listen to the multicast data and control packets from TO and LOs.

An enrolled group user gets information on the multicast session, including IP addresses and port number, via SDP/SAP, HTTP or email.

To ensure that an enrolled user participates in the ECTP connection, the following transport addresses must be announced to the enrolled-group, together with the session-specific information:

1. **Multicast-group transport address**: an IP multicast address and a port number

   The IP multicast address and port number combination must be unique to an ECTP connection, and it will be used by TO to transmit the multicast data to the receivers.

2. **Unicast transport address of TO**: an IP unicast address and a port number

   The IP address and the port number correspond to the source IP address and the port number for the multicast data packets, respectively. They will also be referred to as the destination IP address and the port number for the control packets flowing from receivers to TO.

3. **Unicast transport address of LO**: an IP unicast address and a port number

   This unicast transport address corresponds to the source IP address and the port number for the multicast control packets such as TJ and ACK packets. It is also referred to as the destination address and port number for the control packets from the children to LO.

4. **Multicast control transport address of an LO**: an IP multicast address and a port number

If TO and LOs use different multicast addresses, each LO also announce its multicast address and port numbers. The IP address and port number will be used by LO to transmit the multicast control packets such as HB and RD packets to the children.

Using multicast addresses, each enrolled member must have been attached to the network, via the IGMP and IP multicast routing protocols, before the connection is created. The member listen to the multicast control and data control messages from TO and LOs.

APPENDIX (C)

OPNET PROGRAMMING

OPNET PROGRAMMING

- **OPNET OVERVIEW**

OPNET modular is a network simulation tool developed by OPNET technology INC. It has the following features: (OPNET Technology INC. OPNET Modeling overview, 2003. Available from OPNET Modeler 8.1 software package online documentation).

- **SPECIALIZED IN COMMUNICATION NETWORKS AND INFORMATION SYSTEMS**

  OPNET provides tools to model the detailed functions in data communication technologies and protocol design. OPNET library provides standard protocols such as TCP/IP, RIP and ATM, as well as well-known products such as Cisco routers.

- **OBJECT ORIENTATION**

  Network components are defined in terms of objects, each with set of configurable attributes. Each object is allowed to include other object, if necessary.

- **HIERARCHICAL MODELS**

  OPNET allows users to create network models in hierarchy. For example, users may define a subnet inside a network. OPNET is able to display the higher layer network topology and hid the details of the subnet.

- **SIMULATION TOOLS**

  OPNET allows users to insert probing tools into network models to record statistics such as traffic rate and error rate. Together with its post-simulation analysis tools, OPNET allows users to interpret and evaluate different simulation scenarios.

- **GRAPHICAL USER INTERFACE**

  OPNET has a graphics-based user interface. User may use the GUI to construct a network or a process model, define packet formats, edit object attributes, and visualize simulation results.

- **PROGRAMMING ENVIRONMENT**

  OPNET uses C-styled to modify network object from library or to create new nodes or protocols from scratch. Once the model is created, OPNET automatically integrates and compiles the new model into the runtime environment.

- **NETWORK MODEL**

  A network model specifies the topology of a simulation network. Figure (A) illustrates a typical scenario of a network model. This model is available in OPNET library as IP-firewall.

  This topology shows that a firewall protected local LAN is located at the west cost of USA and there are four clients located at different locations with the US.

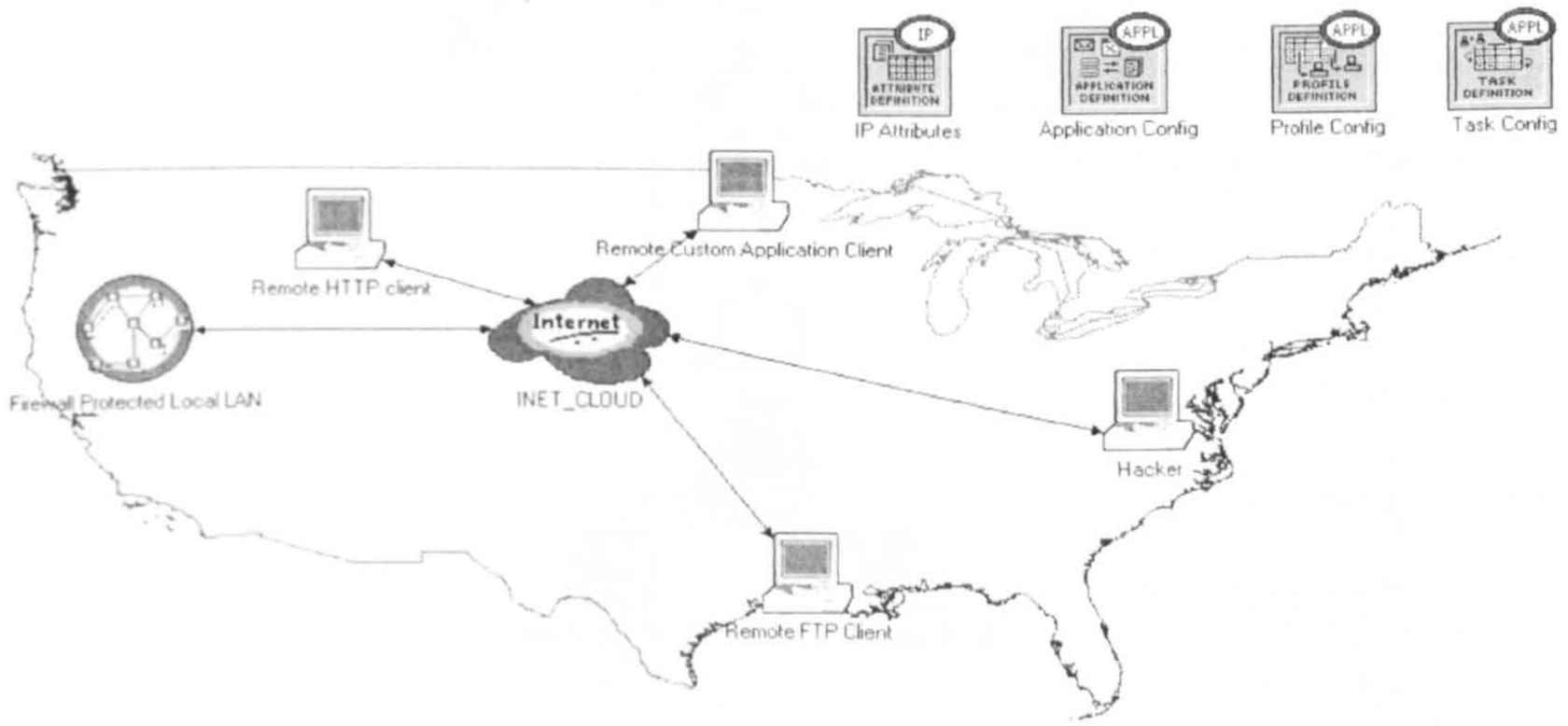


Figure (a): A network model

- **NODE MODEL**

  Node model shows the architecture of node in the network topology. Node model in OPNET represents network nodes or hosts in the simulated network topology. Figure (b) shows the details of a node model. Note that OPNET

organizes the node in such a way that it closely resembles the ISO-ISO layering model: *hub_rx* and *hub_tx* represents the physical interfaces from and to the Ethernet; *mac* and *arp* belong to the data link layer (DLL); IP is modeled op top MAC, with TCP and UDP connected to the IP layer. Finally user application server is modeled on the top of the protocol stack at the application layer.

Each component (such as *ip*, *arp* and *mac*) in the node model is a *process*. A process transport data packets to its neighboring process through *packet streams* (the single arrow solid lines). The arrow shows the direction of the data flow. Each end of the *packet stream* is uniquely numbered in its corresponding process. The single-arrow dash lines in the node model are used to pass statistical data, and thus called statistical lines. Statistical data are used for monitoring process activities. The double arrow dash lines are used to identify logical transmitter and receiver.

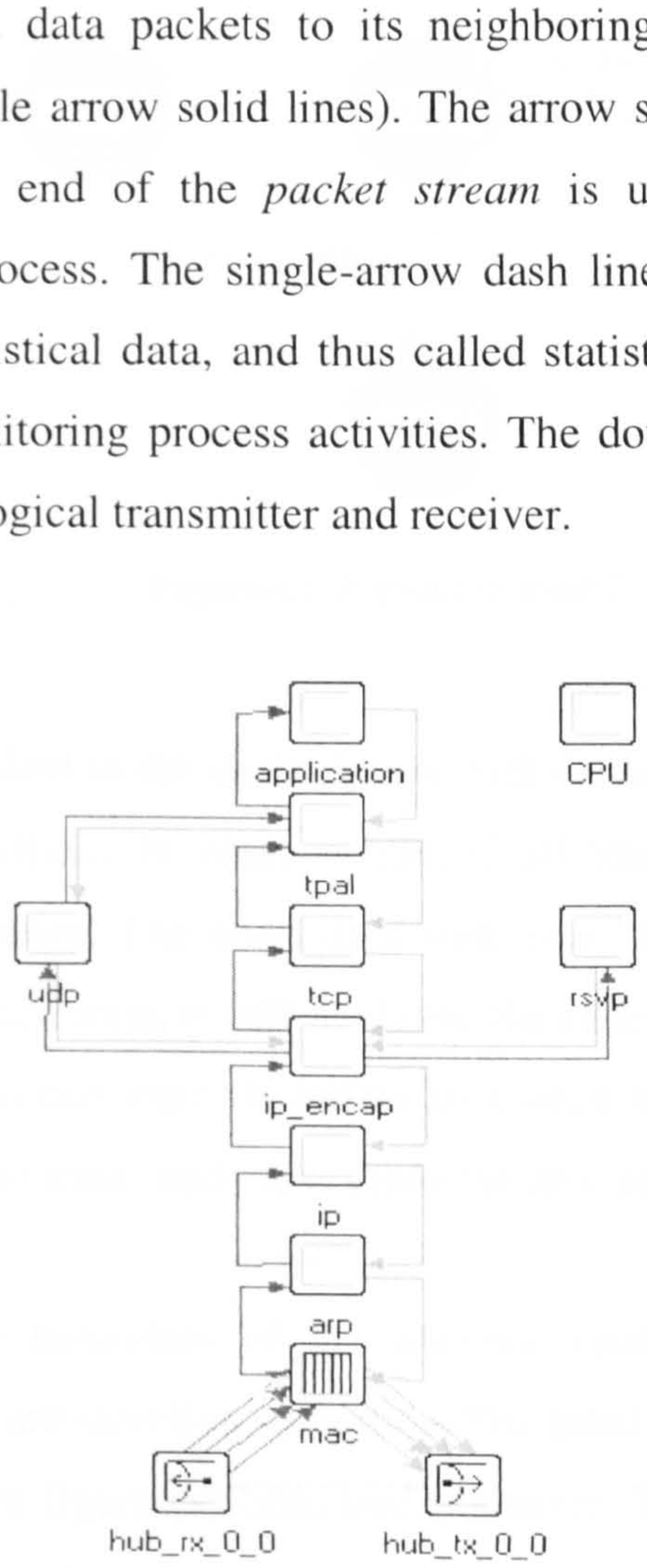
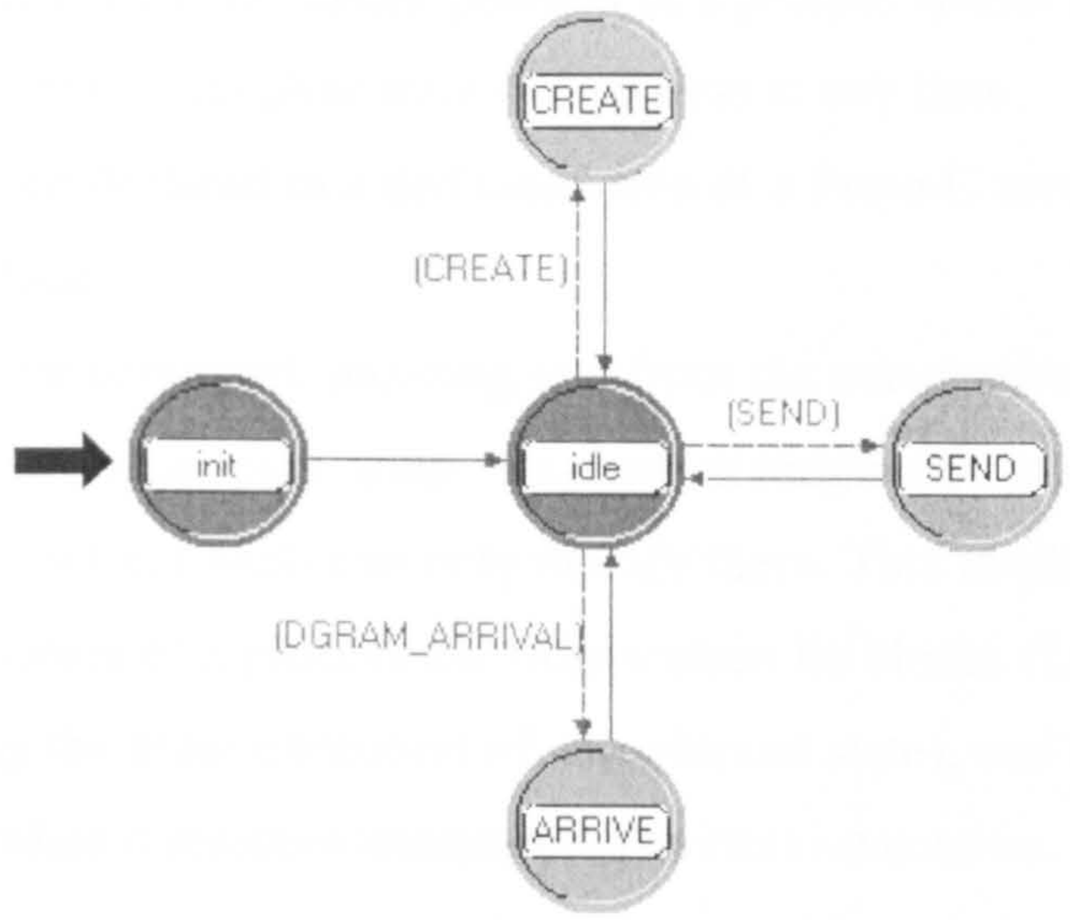


Figure (b): A node model

- **PROCESS MODEL**

  As described in previous section, a process is an important building block for node models. Figure (c) shows the construction of the UDP process mode. OPNET process model defines a starting state (state pointed to by a think arrow),

but does not define a termination state. The links between the states are the transition lines. The labels on the transition lines specify the state transition conditions.



Figure(c): A process model

The solid line is equivalent to the dashed lines with the label "default". They indicate an unconditioned transition. In other words, if all transition conditions fail, this (default) transition is taken. The solid dark state is a "unforced" which means that once a process enters such state, it will not leave the state until an event happens. The solid which state is "forced state. It means that once the process enters the state, executes the script in the state, and moves into the new state without waiting for their events to take place.

A process models the behaviors of the network protocol or an application. C-language-styled scripts are stored in each state. The label of a state partitions the state (represented by circle in figure (a)) into two segments. The top segment *State Enter Executive* (enter-exec), and the bottom segment represents *State Exit Executive* (exit-exec). Scripts in the enter-exec and the exit-exec segments are executed on entering or exiting the state, respectively. Not all states require having both entered exit execution defined. For example,, the forced states usually do not need an exit execution because there is no state change between the execution of enter-exec and exit-exec. For unforced states, it may not be the case because the process may need to process the never that triggers the state transition. Functions or statements not related to a particulate state are organized as follows:

- **STATE VARIABLES**

  Variables that are used to represent the information accumulated and retained by a process are referred to as state variables. This name is due to the fact that these variables, together with the current position of a process within its state transition diagram, represent the complete state of a process at any time.

  State variables are declared in a dedicated area of a Proto-C model, referred to as state variable block.

  State variables are persistent, meaning that from the perspective of each process, they retain their value over time. Explicit assuagement statements that are executed by the process itself can only modify them. This implies that the values of the state variables of a process are frozen when its blocks (i.e. halts execution after completing the enter execution of an unforced state), and found unchanged by the process when it resumes execution at its next invocation.

- **TEMPORARY VARIABLES**

  A process may declare temporary variables in order to store information that does not require persistence. Temporary variables are not guaranteed have any particular set of values when a process model is invoked. In particular, the values that were assigned to these variables by the same process during its previous invocation may have been changed.

  Since they have no persistence, temporary variables are useful only as scratch storage to facilitate computation of complex expression or to hold information within the time frame of a single process invocation.

- **GLOBAL VARIABLES**

  Global variables provide a means for multiple processes in different modules, including processes of different types, to store information in a common location. Declarations should be entered into a process model's header block (HB).

- **FUNCTION BLOCK (FB)**

  In Proto-C, any number of functions may be specified within a section called function black of a process model. This function black's contents can be edited by the using edit function block operation of process editor.

The most common and straightforward approach to modularizing recurring computations is to make use of the function call mechanism provided by the C and C++ languages.

Functions (or procedures) are specifications of computations that may accept parameters, referred to as arguments. Arguments may be of any data type provided by C, C++ or OPNET.

Depending on the service that is required of them, functions may provide returns values that results from their computations or simply perform a series of actions and return no results.

If functions return values that are not integers, these functions should be declared prior to their use. If they are used within the STDs executive and condition statements, corresponding function declarations should be placed in the header block, to ensure that they appear in the proper order.

The purpose of a function declaration is to inform the compiler what type of value will be returned.

- **DIAGNOSTIC BLOCK (DB)**

OPNET simulations automatically incorporate a sophisticated "debugger" called OPNET Diagnostic Block (ODB) that allows the user to interactively monitor the progress of a simulation and the state of its object. ODB's capabilities are exercised from a command line. Most of these capabilities are entirely provided by the simulation kernel. However, because processes are user-defined and can store information in propriety data structures, and because the simulation kernel is not capable of knowing which particular information within a process is interesting to report, ODB provides two special commands that can be applied to processes. These commands, called **`prodiag`** and **`proldiag`**, allow useful reports to be printed about the current sate of a user-defined process.

- **Termination block**

Procedures in termination block is invoked when the simulation terminations. For example, one may create a function in the termination block that collects all the statistics. At the time the simulation terminations, OPNET automatically executes the function in the termination block to collect final results.

As mentioned above, once the process enters an " unforced" satet, it must wait for an event before it can move to the next state, such as stream interrupt, self-interrupt, and remote interrupt, and so on. Once the process receives an interrupt, the interrupt type can be found by calling a function op_intrpt_type(). The return value indicates the type of interrupt.

Stream interrupt happens when the process completely receives a data deliver by neighboring process. The packet stream interface can be founded by calling op_intrpt_stm().

Self-interrupt is generated by the process itself. The process schedules a self-interrupt for a future simulation time. A different process in the same node generates remote interrupt.

- **SIMULATION ATTRIBUTES**

  In order to avoid hard-code simulation variables such as IP addresses in the simulation program, simulation attributes are used as an interface for users to interact with the simulation. Each object, e.g. process model, node model and the network model, can define simulation attributes. The simulation attributes are set before the simulation states. During the simulation, the process reads the simulation attributes during the initialization state so that critical information can be established before data pass through.

- **SIMULATION RESULTS**

  During the execution of the simulation, OPNET collects statistics of common interest, for example, delay for voice applications. Users can also define their own statistics and let OPNET collect the data during the simulation. The left side of figure (a) shows the interface that displays the available statistics within a node (or link) model, and the right side of figure (a) shows a plot of selected statistics after the simulation is completed.

- **DEBUG**

  In order to improve performance, OPNET kernel runs in the optimized mode. However, the kernel mode can be switched to the debugging mode. OPNET provides its own debugging tools, which are different from other popular debugger such as GDB. OPNET debugger does not allow the user to trace the

source code and execute the program instruction by instruction. Instead, it debugs the simulation one event at a time. Inserting the following code block into the program can print debugging information:

```
If (op_prg_odb_ltrace_enbale ("lable_string")){
    op_prg_odb_print_major("string1", "string2",...,NIL);
}
```

The function op_prg_odb_ltrace_enbale () returns FALSE if the label string is not registered in the simulation environment. It returns TRUE otherwise. User can register the label string in the simulation environment by issuing the following command at the debug prompt: lstop "label_trace". User can also remove a label string from the simulation environment by using the following command at the debug prompt: dellstop<break_ponitID>.

The function op_prg_odb_print_major () prints out strings generated by the user, together with the simulation information such as the simulation time, an event received before entering or exiting the state, and the event the simulation kernel is going to issue.

- **CONFIGURATION APPLICATIONS AND PROFILES**

To configure a workstation or LAN to model the behavior of a user or group of users, it is needed to describe their behavior. A user 's behavior or 'profile' can be described by the applications.